VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

APLIKACE PRO VIZUALIZACI OBRAZŮ V ROZŠÍŘENÉ REALITĚ NA IOS

IOS APPLICATION FOR PAINTING VISUALIZATION IN AUGMENTED REALITY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **SAMUEL MENSÁK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. PETR BOBÁK,**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Mensák Samuel**
Program: Informační technologie
Název: **Aplikace pro vizualizaci obrazů v rozšířené realitě na iOS**
iOS Application for Painting Visualization in Augmented Reality
Kategorie: Uživatelská rozhraní

Zadání:

1. Seznamte se s architekturou iOS, jazykem Swift a problematikou rozšířené reality. Nastudujte designové principy návrhu uživatelského rozhraní pro platformu iOS.
2. Analyzujte možnosti knihovny ARKit a prozkoumejte podobné existující aplikace související s vizualizací objektů v rozšířené realitě.
3. Prozkoumejte architekturu klient-server včetně nástrojů vhodných pro tvorbu webových služeb.
4. Navrhněte systém typu klient-server pro podporu prodeje obrazů. Uživatelské rozhraní klientské i webové aplikace navrhněte iterativním způsobem.
5. Navržený systém implementujte. V rámci klientské aplikace se zaměřte na realistické zobrazení obrazů v rozšířené realitě.
6. Otestujte funkcionalitu a použitelnost výsledné aplikace.
7. Zhodnoťte dosažené výsledky, vytvořte plakát a krátké prezentační video. Dále navrhněte možné pokračovaní.

Literatura:

- ARKit: https://developer.apple.com/documentation/arkit
- Human Interface Guidelines: https://developer.apple.com/design/human-interface-guidelines/
- dále dle pokynů vedoucího

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Bobák Petr, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 6. dubna 2021

Abstrakt

Táto bakalárska práca sa zaoberá rozšírenou realitou na mobilných zariadeniach spoločnosti Apple so zameraním na vizualizáciu obrazov na vertikálnych plochách. Na začiatku práce sú popísané technológie spojené s rozšírenou realitou, vývojom mobilných a serverových aplikácií v programovacom jazyku Swift. Na týchto technológiách je následne založený návrh a implementácia výslednej aplikácie a prislúchajúceho webového serveru. Práca popisuje celý proces vývoja od návrhu užívateľského rozhrania, cez implementáciu a testovanie, až po výslednú aplikáciu a webový server. Výsledkom je systém klient-server umožňujúci zobrazovanie obrazov v rozšírenej realite pomocou mobilnej aplikácie.

Abstract

This bachelor thesis deals with augmented reality on mobile devices made by Apple with focus on visualization of images on vertical plane. At the beginning of the thesis, reader is informed about technologies related to augmented reality and development of mobile and web applications in Swift programing language. Based on these technologies, mobile application and web server is designed and implemented. The thesis describes whole process of development starting with designing of UI, continuing with implementation and testing and finishes with working mobile application and web server. The resulting client-server system enables user to visualize paintings in augmented reality using mobile app.

Kľúčové slová

rozšírená realita, iOS, Swift, ARKit, Vapor, mobilné aplikácie, server, klient-server

Keywords

augmented reality, iOS, Swift, ARKit, Vapor, mobile applications, server, client-server

Citácia

MENSÁK, Samuel. *Aplikace pro vizualizaci obrazů v rozšířené realitě na iOS*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Petr Bobák,

Aplikace pro vizualizaci obrazů v rozšířené realitě na iOS

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Petra Bobáka. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Samuel Mensák
11. mája 2021

Poďakovanie

Poďakovať by som sa chcel vedúcemu tejto bakalárskej práce, Ing. Petrovi Bobákovi, za podnetné pripomienky a usmernenia ktoré ma pomohli nasmerovať tým správnym smerom. Moja vďaka patrí ale aj všetkým ostatným ktorý ma pri písaní tejto práce podporovali.

Obsah

Kapitola 1

Úvod

Táto bakalárska práca má za cieľ vytvoriť systém typu klient-server pre podporu predaja obrazov. V súčastnosti existujú rôzne technológie a platformy na ktorých je možné pracovať s rozšírenou realitou. Aplikácia ktorá je výsledkom tejto práce bude určená pre zariadenia od spoločnosti Apple a teda operačný systém iOS.

Koncept rozšírenej reality nieje novinkou, prvé zariadenia využívajúce tento koncept sa objavili v roku 1940 v britských bojových lietadlách v podobe HUD[1]. Prvé použitie rozšírenej reality na mobilnom telefóne bolo predvedené skupinou pozostávajúcou z Mathiasa Möhringa, Christiana Lessiga a Olivera Bimbera v roku 2004.[18] Avšak len v posledných rokoch sa technológia rozšírenej reality dostala do povedomia bežných ľudí. To môžme pripísať rozšíreniu mobilných zariadení typu smartfón či tablet medzi užívateľov a taktiež aj technologickému pokroku daných zariadení, ktorý umožnil širšie využitie rozšírenej reality.

V úvodnej kapitole 2 sa táto práca venuje detailnejšiemu popisu rozšírenej reality. Nachádza sa tu stručné zhrnutie vývoja rozšírenej reality či súčastného stavu a možností tejto technológie. V tejto kapitole sú taktiež čitateľovi priblížené technológie umožňujúce vývoj mobilných aplikácii využívajúcich rozšírenú realitu na platformách ako ARCore 2.3 pre Android či ARKit 3 pre iOS respektíve iPadOS. Nachádza sa tu aj prehľad najpopulárnejších aplikácii využívajúcich rozšírenú realitu.

V kapitolách o návrhu 4 a implementácii 5 sú priblížené nasledovné témy: platforma iOS (iPadOS) z hľadiska vývoja aplikácií a technológie potrebné na návrh a implementáciu serveru, za pomoci programovacieho jazyku Swift a jeho webových frameworkov ako napríklad Vapor. Ďalej sa práca venuje popisu frameworku ARKit ktorý na platforme iOS umožňuje vývoj aplikácií pracujúcich s rozšírenou realitou. Rovnako tu čitateľ môže nájsť špecifikáciu požiadavkou na server a aplikáciu. Medzi ne možno zaradiť správu databázy obrazov, možnosť interakcie s virtuálnymi objektami, realistické zobrazenie obrazov z hľadiska veľkosti, presného umiestnenia v priestore a iné. Spôsob akým sú tieto požiadavky realizované vo výslednej implementácií je popísaný v implementačnej časti.

V záverečnej kapitole 6 práca obsahuje informácie o testovaní výslednej aplikácie a webového serveru. Aplikácia a server boli testované z dvoch hľadísk, funkčného – či a ako dobre umožňujú vykonávať funkcie ktoré boli definované pri návrhu a užívateľského – ako sú tieto funkcie prístupné užívateľovi. Nachádza sa tu aj prehľad vlastností či funkcii ktoré by mohli byť v budúcnosti do aplikácie alebo serveru doplnené.

[1] Head-up display

Kapitola 2

Rozšírená realita

Rozšírenú realitu (augumented reality) môžme definovať ako prepojenie skutočného prostredia a digitálne vytvorených informácií ako obrázky, video, zvuk či hmatová odozva v reálnom čase. Na rozdiel od virtuálnej reality ktorá skutočný svet celkom nahrádza, rozšírená realita len reálne prostredie doplňuje.[15]

V tejto kapitole sa práca detailnejšie zaoberá historickým vývojom rozšírenej reality, súčasnosťou, popisom technológií ktoré umožňujú vývoj aplikácií pracujúcich s rozšírenou realitou a prehľadom aplikácií ktoré sú v tejto oblasti v súčastnosti dostupné a populárne.

2.1 História

Jednu z prvých zmienok o technológií podobnej rozšírenej realite môžme nájsť v knihe The Master Key: An Electrical Fairy Tale od L. Franka Bauma z roku 1901 kde autor popisuje akési okuliare ktoré dokážu tomu kto ich používa odhaliť skryté charakterové vlastnosti okolitých osôb.[9]

Prvé reálne využitie nachádza rozšírená realita v roku 1940 v armáde. Jedná sa o HUD pre britské bojové lietadlá určené k nočným letom. Prvé nositeľ né zariadenie sa objavuje v USA začiatkom 60tých rokov. V 90tých rokoch sa objavuje rozšírená realita v televízií v podobe pridaných informácií počas priamych športových prenosov. Prvé komerčne dostupné okuliare, vyobrazené na obrázku 2.1, využívajúce rozšírenú realitu predstavuje firma Sony v roku 1996.[18]

V roku 2004 je po prvý krát demonštrované použitie rozšírenej reality na mobilnom zariadení tímom z Bauhaus University. V roku 2017 spoločnosť Apple predstavuje framework ARKit, ktorý vývojárom značne uľahčuje vývoj aplikácií využívajúcich rozšírenú realitu pre operačné systém iOS a o rok neskôr s podobným riešením prichádza aj Google pre operačný systém Android.

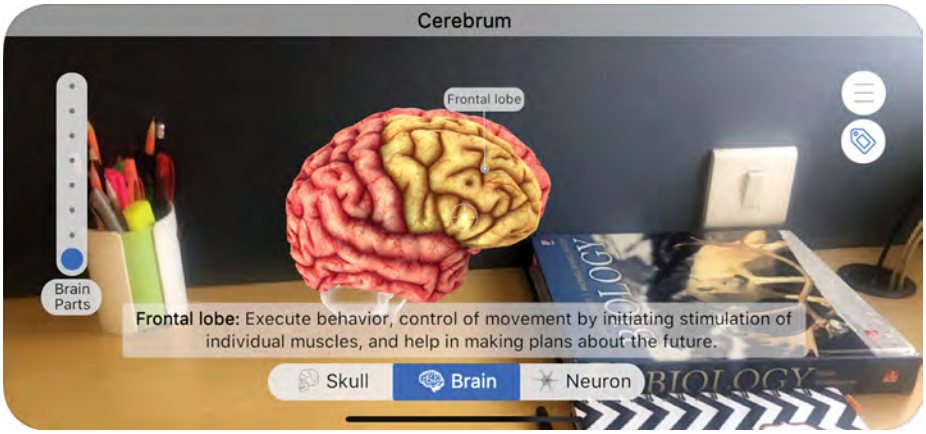
2.2 Súčasnosť a využitie

V súčastnosti sa rozšírená realita rozširuje do viacerých odvetví. Mimo už tradičného použitia v armáde a letectve môžme sledovať zvýšenú popularitu medzi bežnou populáciou. Toto možno pripísať rozšíreniu zariadení podporujúcich prácu s rozšírenou realitou s tým spojenému nárastu počtu AR aplikácií.

Obr. 2.1: Sony Glasstron vydaný v roku 1996. Zariadenie obsahuje zabudované slúchadlá a 2 LCD obrazovky o veľkosti 0.55 palca s mechanickou uzávierkou ktorá umožňovala priehľadnosť.[11]

Vzdelávanie

Rozšírená realita si nachádza čoraz väčšie uplatnenie vo vzdelávacom systéme. Umožňuje do výuky pridať interaktivitu, ktorá učenie spraví atraktívnym a v mnohých prípadoch značne uľahčí či sprehľadní. Príkladom sú aplikácie pre mobilné zariadenia ktoré zobrazujú preberané predmety prostredníctvom mobilného zariadenia priamo v priestore pred žiakom a umožňujú mu s nimi prostredníctvom daného mobilného zariadenia aj manipulovať. Príklad mobilnej aplikácie určenej pre vzdelávanie je zobrazený na obrázku 2.2.



Obr. 2.2: Aplikácia Brainapse. Aplikácia umožňuje štúdium anatómie ľudského mozgu prostredníctvom rozšírenej reality. Prvky užívateľského rozhrania umožňujú navigovanie a manipuláciu zo zobrazeným obsahom.[10]

Priemysel

V priemysle si rozšírená realita nachádza miesto pri asistencií vo výrobe. Jedná sa o mobilné aplikácie pracujúce s rozšírenou realitou alebo kombináciu kamier a projektorov ktoré doplňujú realitu bez nutnosti použitia mobilného zariadenia či špeciálnych okuliarov.

Typickým príkladom sú mobilné aplikácie ktoré umožňujú programovanie robotických ramien v rozšírenej realite. Ich nespornou výhodou je fakt, že operátor vidí pohyb ramena v priestore a môže opraviť prípadné nebezpečné/nežiadúce pohyby ešte pred tým než sa rameno reálne pohne. Rozšírená realita nachádza uplatnenie aj pri opravách či údržbe kedy pracovníkovi asistuje pri identifikácií častí zariadenia prípadne ich porúch. Táto funkcionalita AR aplikácií je demonštrovaná obrázkom 2.3.

Obr. 2.3: Ukážka aplikácie ktorá rozoznáva a zvýrazňuje časti zariadenia. Používa sa pri kontrole a opravách zariadení. [17]

Zábava

V posledných rokoch pre bežného používateľa zrejeme najviditeľnejšia forma využitia rozšírenej reality. Do tejto kategórie môžme zaradiť hry v rozšírenej realite, aplikácie ktoré v AR zobrazujú príbehy, aplikácie ktoré pridávajú v reálnom čase do videí objekty či upravujú alebo menia vzhľad používateľa. Obrázok 2.4 zobrazuje príklad hernej mobilnej aplikácie pracujúcej s rozšírenou realitou.

Obr. 2.4: Adaptácia populárnej hry Minecraft od štúdia Mojang do rozšírenej reality pre mobilné zariadenia.[1]

Zdravotníctvo

Rozšírená realita v zdravotníctve môžem mať rôzne podoby. Jedným z možných použití je možnosť interaktívneho prezerania si virtuálneho ľudského tela, čím rozšírená realita umožňuje napríklad rýchlejšie vzdelávanie budúcich doktorov či lepšie naplánovanie zákroku. Znázornené na obrázku 2.5. Ďalšou možnosťou použitia rozšírenej reality v zdravotníctve je pridávanie informácii o pacientovi resp. tom čomu sa lekár venuje priamo počas zákroku za využitia AR okuliarov. Táto technológia značne pomáha s orientáciou pri komplikovaných zákrokoch a znižuje riziko chyby. [2]

Obr. 2.5: Príklad využitia rozšírenej reality v zdravotníctve. Za pomoci AR okuliarov si lekári môžu prezerať 3D model operovanej časti aj počas zákroku.[12]

2.3 Platformy na vývoj AR aplikácií

V súčastnosti je na trhu pomerne široký výber frameworkou ktoré umožňujú prácu s rozšírenou realitou. Táto práca sa zameriava na tie z nich, ktoré umožňujú pracovať s rozšírenou realitou na mobilných zariadeniach. Jedná sa spravidla o riešenia pre konkrétnu platformu ale dostupné sú aj mnohé multiplatformové riešenia.

ARKit

ARKit je framework umožňujúci vývoj aplikácií v rozšírenej realite od spoločnosti Apple. Predstavený bol v roku 2017 na WWDC[1].[7] Podporovaný je vývoj pre operačné systémy iOS a iPadOS. ARKit podporujú len zariadenia s systémom iOS 11 a novším, respektíve iPadOS 13 a novším. Vyžadovaný je taktiež SoC[2] A9 alebo novší. Detailnejšie sa práca tomuto frameworku venuje v sekcií 3.3.

ARCore

ARCore je platforma pre vývoj mobilných aplikácií v rozšírenej realite od spoločnosti Google. Medzi jej hlavnú funkcionalitu patrí sledovanie polohy mobilného zariadenia, detekcia okolitých povrchov a odhad svetelných podmienok v okolitom reálnom prostredí. ARCore

[1] Worldwide Developers Conference
[2] system on chip

za splnenia istých podmienok umožňuje aj interakciu s aplikáciami vytvorenými pre konkurenčný framework ARKit. Táto funkcionalita je zabezpečená zdieľaním vybraných bodov zo scény prostredníctvom špecializovaného serveru. Pre svoje fungovanie ARCore vyžaduje operačný systém Android vo verzii minimálne 7.0 a kompatibilné zariadenie.[4] Framework ARCore bol predstavený v roku 2018. Príklad aplikácie využívajúcej ARCore je zobrazený na obrázku 2.6.

Obr. 2.6: Ukážka z aplikácie Pokemon GO ktorá využíva framework ARCore. Jedná sa v súčastnosti o zrejme najpopulárnejšiu hru v rozšírenej realite. [5]

Vuforia

Jedná sa o najrozšírenejší multiplatformový framework na vývoj mobilných aplikácií pracujúcich s rozšírenou realitou. Vuforia je vyvíjaná spoločnosťou PTC a jej použitie nieje spoplatnené. Podporované sú operačné systémy Android, iOS (iPadOS) a framework umožňuje vývoj aplikácií aj pre UWP[3] od Microsoftu. Príklad mobilnej aplikácie vytvorenej pomocou Vuforia frameworku sa nachádza na obrázku 2.7. Framework ponúka, až na zanedbateľné rozdiely, ekvivalentnú funkcionalitu ako vyššie spomínané konkurenčné riešenia ARKit a ARCore. Podporované je rozoznávanie 2D a 3D objektov a navyše aj použitie externej kamery.[26]

Obr. 2.7: Ukážka použitia mobilnej aplikácie ReBlink ktorá využíva framework Vuforia na zariadení s operačným systémom iOS. Aplikácia kombinuje umenie (obrazy) a rozšírenú realitu v zábavnej forme.[21]

[3] universal windows platform

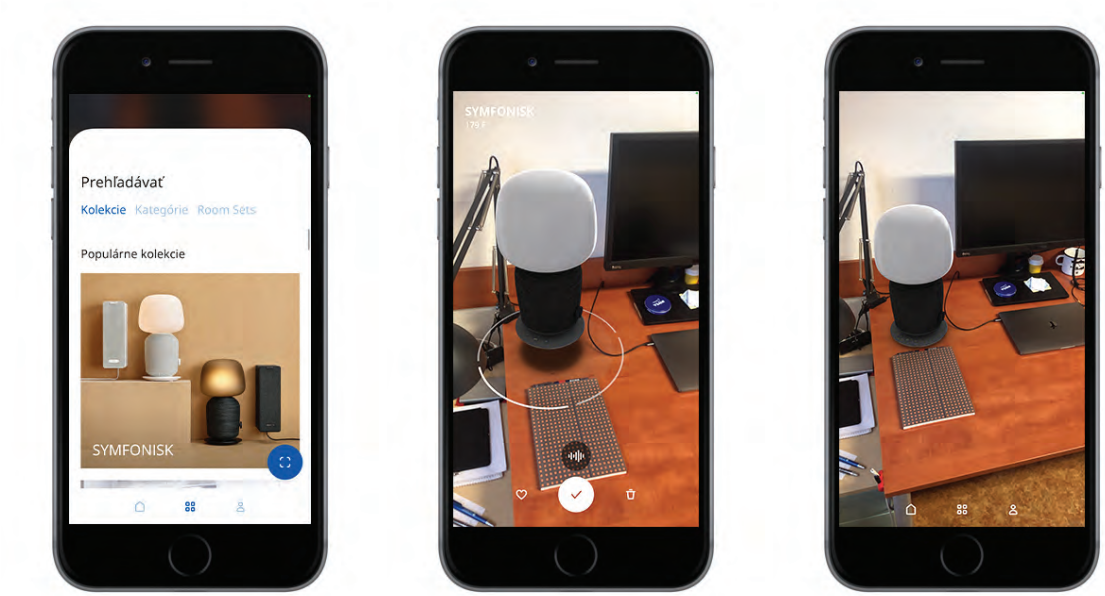
2.4 Existujúce aplikácie

Táto časť práce je venovaná popisu a analýze existujúcich mobilných AR aplikácií na platforme iOS. Nakoľko týchto aplikácií existuje množstvo, analýza sa venuje len výberu tých najpopulárnejších a tých ktoré sú svojou funkcionalitou podobné aplikácií ktorou sa zaoberá táto bakalárska práca. Cieľom analýzy je preskúmať existujúce riešenia, ich funkcionalitu či trendy ktoré sa týkajú používateľského rozhrania AR aplikácií. V neposlednom rade sa analýza zaoberá aj nedostatkami či chybami existujúcich riešení.

IKEA Place

Aplikácia IKEA Place slúži na vizualizáciu vybraných produktov spoločnosti IKEA v rozšírenej realite. V súčastnosti aplikácia ponúka na zobrazenie výber z vyše 2200 produktov tejto spoločnosti. Produkty sú zobrazované v reálnej veľkosti na horizontálnych plochách.[13]

Jedná sa o jednu z prvých skutočne rozšírených aplikácii tohto typu s veľkou popularitou a pozitívnou odozvou od používateľov. IKEA Place ponúka prehľadné užívateľské rozhranie ktoré je dobre členené. Jedinou vážnejšou nevýhodou aplikácie je nemožnosť resetovať AR scénu a teda nutnosť nežiadúce objekty odstraňovať postupne. Užívateľské rozhranie aplikácie je zobrazené na obrázku 2.8.

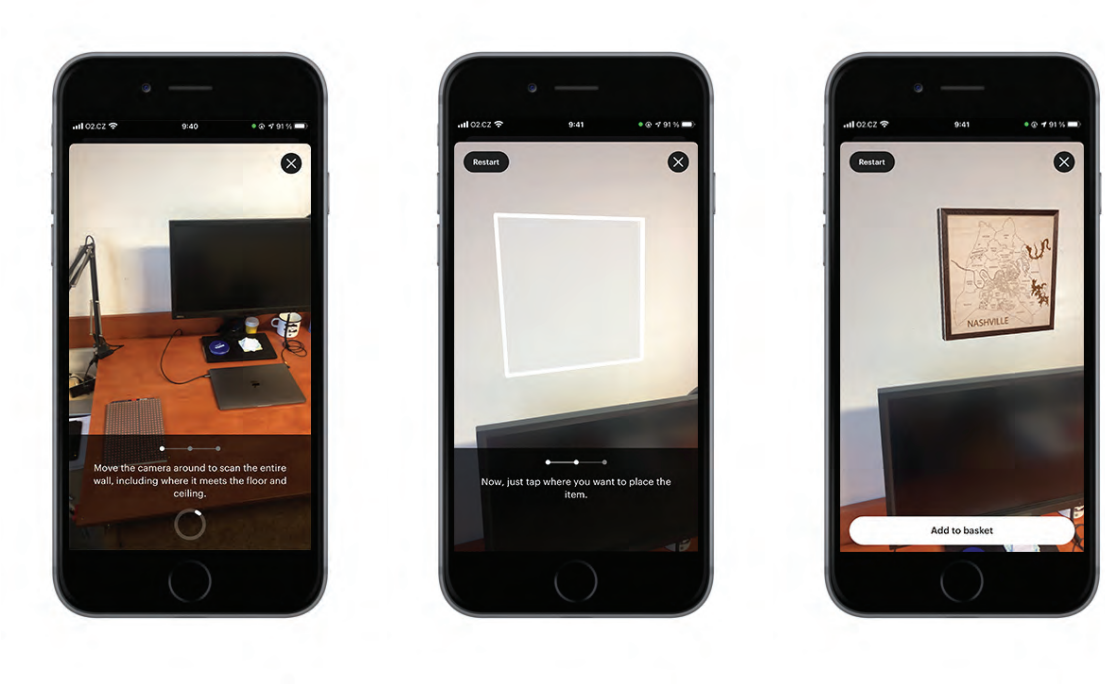


Obr. 2.8: Ukážka používateľského rozhrania mobilnej aplikácie IKEA Place. Zľava: výber objektu z ponuky, interakcia s objektom umiestneným v scéne a výsledná scéna v rozšírenej realite.

Etsy

Jedná sa o aplikáciu rovnomenného internetového obchodu, ktorá umožňuje užívateľom pri určitom druhu tovaru jeho vizualizáciu v rozšírenej realite. Ide zväčša o obrazy prípadne iné nástenné dekorácie.

Funkcionalita tejto mobilnej aplikácie ktorá využíva rozšírenú realitu sa s pomedzi preskúmaných aplikácií najviac blíži funkcionalite ktorú má za cieľ implementovať výsledná aplikácia z tejto bakalárskej práce. Nakoľko je však bola do aplikácie rozšírená realita pridaná len relatívne nedávno, jej možnosti sú značne obmedzené. V čase testovania umožňovala mobilná aplikácia len umiestnenie objektu na vertikálnu plochu v zobrazovanom priestore a prípadnú zmenu jeho veľkosti. Užívateľské rozhranie aplikácie od Etsy je zobrazené na obrázku 2.9.



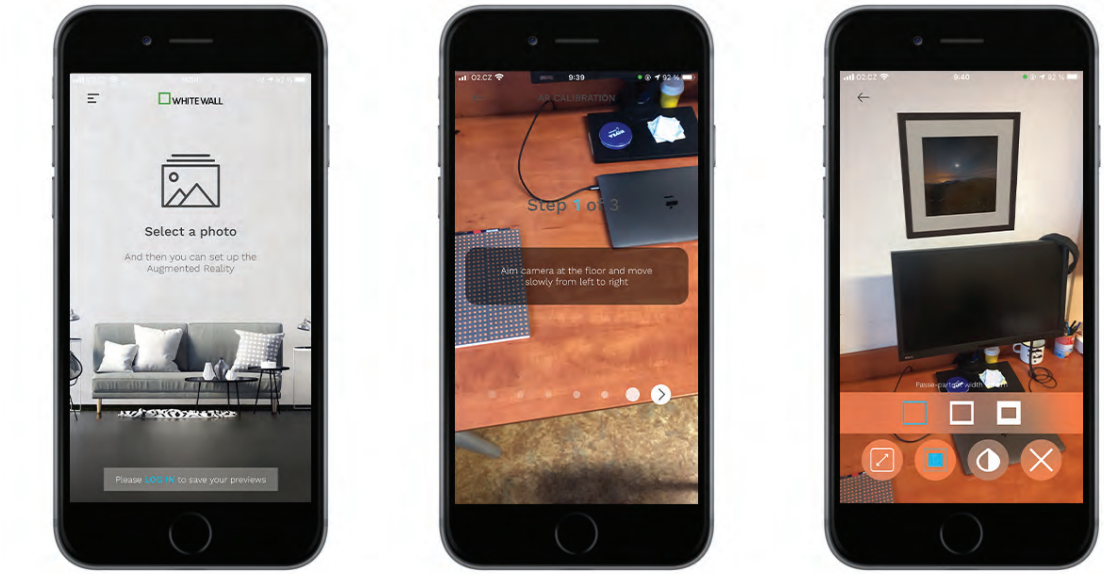
Obr. 2.9: Ukážka používateľského rozhrania mobilnej aplikácie od Etsy. Zľava: detekcia vertikálnych plôch v okolí zariadenia s cieľom vytvoriť scénu rozšírenej reality, výber umiestnenia objektu na vertikálnej ploche v scéne a zobrazenie objektu (obrazu) v AR scéne na vybranom mieste.

White Wall AR

Mobilná aplikácia White Wall AR od rovnomennej, fotky tlačiacej, spoločnosti White Wall slúži k vizualizácií obrazov nielen z jej ponuky. Aplikácia White Wall AR umožňuje aj vizualizáciu ľubovolnej fotografie zo zariadenia za použitia predom definovaných rámov, materiálov a veľkostí obrazov.

Jedná sa o aplikáciu ktorá v rozšírenej realite pracuje len s vertikálnymi plochami a ponúka plynulý používateľský zážitok a intuitívne ovládanie. White Wall AR taktiež vybočuje s radu podobných aplikácií netradičným riešením výberu iniciálnej polohy umiestňovaného obrazu. Užívateľovi stačí pridržať zariadenie na požadovanom mieste v reálnom svete a

aplikácia na toto miesto následne umiestni daný obraz. Jediným negatívom je limitovaná možnosť zmeny vizualizovaných obrazov. Detail užívateľského rozhrania aplikácie White Wall AR je zobrazený na obrázku 2.10.



Obr. 2.10: Ukážka používateľského rozhrania aplikácie White Wall AR. Zľava: výber fotografie na zobrazenie v AR scéne, inicializácia AR scény a možnosti zmeny veľkosti objektu.

Kapitola 3

Swift, Server-side Swift a ARKit

V tejto kapitole sa venujem technológiám ktoré umožňujú vývoj mobilných aplikácií pracujúcich s rozšírenou realitou na zariadeniach spoločnosti Apple a tiež tým ktoré umožňujú jeho použitie pre tvorbu webového serveru.

V prvej časti tejto kapitoly je detailnejšie popísaný programovací jazyku Swift v ktorom bude implementovaná výsledná aplikácia a webový server. V druhej časti sa venujem popisu frameworkou pre implementáciu serverovej a aplikačnej časti za použitia programovacieho jazyku Swift. Z serverovo orientovaných frameworkou sa táto kapitola venuje trojici, konkrétne frameworkom Kitura, Perfect a Vapor. Ako framework pre prácu s rozšírenou realitou je popísaný framework ARKit, od spoločnosti Apple ktorý umožňuje prácu s AR na cieľovej platforme iOS respektíve iPadOS.

3.1 Programovací jazyk Swift

Swift je moderný programovací jazyk vyvíjaný spoločnosťou Apple. Predchodcom, ktorého Swift postupne nahrádza je Objective C. Predstavený bol v roku 2014 na WWDC. Verzia 2 prišla o rok neskôr, opäť na WWDC. S verziou 2.2 prišla zmena v podobe sprístupnenia jazyku ako open-source. Vo verzií 3 prišlo k podstatnému vylepšeniu syntaxe programovacieho jazyka. Začiatkom roku 2018 Swift predbehol v obľúbenosti predchádzajúci programovací jazyk používaný pre vývoj na platformách Apple, Objective C. Posledná verzia v čase písania práce je 5.4. Jazyk umožňuje vývoj na všetkých platformách spoločnosti Apple (iOS, iPadOS, watchOS, tvOS, MacOS) a je podporovaný aj na platformách Linux (od verzie 2.2) a Windows (od verzie 5.3).

Cieľom jazyku je poskytnúť modernú, prehľadnú syntax a vysokú bezpečnosť výsledného kódu. Bezpečnosť je docielená napríklad vynútenou inicializáciou premenných, nemožnosťou priradenia typu `nil` akémukoľvek objektu (pokus o priradenie štandardne vedie k chybe počas prekladu) a jasnému rozlíšeniu premenných a konštánt na úrovni syntaxe kedy premenné využívajú kľúčové slovo `var` a konštanty `let`. Použite programovacieho jazyku Swift na platformách od Apple umožňuje, vďaka Objective C runtime knižnici, v rámci jedného programu písať a spustiť kód v programovacích jazykoch Swift, C, C++ a Objective C.

Programovací jazyk Swift pre správu pamäte využíva mechanizmus ARC[1]. Na rozdiel od Objective C, tento prístup nevyžaduje manuálny manažment pamäte programátorom ale prináša iné potencionálne problémy. Najvážnejší je vznik silného referenčného cyklu, kedy sa dva alebo viaceré objekty odkazujú navzájom a neumožňujú svoje uvoľnenie z pamäte

[1] Automatic Reference Counting

aj keď na nich žiadna externá referencia neexistuje. Apple sa snaží tento problém v Swifte redukovať poskytnutím možností nastaviť referenciu slabého respektíve nevlastneného typu, typicky používanú pri vzťahoch dieťa-rodič. Vzhľadom na to, že programovací jazyk Swift je open source, naskytá sa možnosť jeho využitia aj pre web. Už existujúce frameworky, ktoré webové použitie Swiftu umožňujú, práca detailnejšie popisuje v nasledujúcej sekcií.[22]

3.2 Server side Swift

Kitura

Framework Kitura je open-source webový framework, licencovaný pod Apache License 2.0 napísaný v programovacom jazyku Swift ktorý výsledkom vývoja realizovaného spoločnosti IBM. V decembri roku 2019 spoločnosť IBM ohlásila, že nieje v jej záujme ďalej pokračovať vo vývoji a v januári 2020 bol vývoj zo strany IBM zastavený. Framework v súčastnosti pomerne úspešne prešiel pod správu komunity. Poslednou oficiálnou verziou je verzia 2.9.1 z novembra 2019.

Kitura framework podporuje štandardnú funkcionalitu očakávanú od webového frameworku, akou je napríklad URL smerovanie (GET, POST, PUT, DELETE a iné), URL parametre, parsovanie JSON, statické poskytovanie súborov, podporu pre SSL/TLS, integráciu s SQL (PostgreSQL, MySQL a SQLite) a NoSQL databázami (CouchDB, IBM Cloudant a MongoDB) či použitie šablón typu Stencil alebo Markdown.

Zaujímavosťou tohto framworku je možnosť použitia na platformách IBM Z a x86 pod operačnými systémami MacOS a Linux.[16]

Perfect

Perfect je framework pre aplikačné servery, weby a servery implementovaný vo Swifte. Jedná sa o open-source projekt pod licenciou Apache. Stojí za ním spoločnosť PerfectlySoft. Verzia 1.0 bola vydaná v novembri 2015, tesne pred tým než sa programovací jazyk Swift stal open-source. Framework stavia na základoch programovacieho jazyka Lasso. Od verzie 2.0 podporuje modularitu a nenúti vývojárov do projektov obsiahnuť všetky svoje funkcie čo znižuje nároky na pamäť. Podporovaná je platforma x86 a operačné systémy MacOS a Linux.

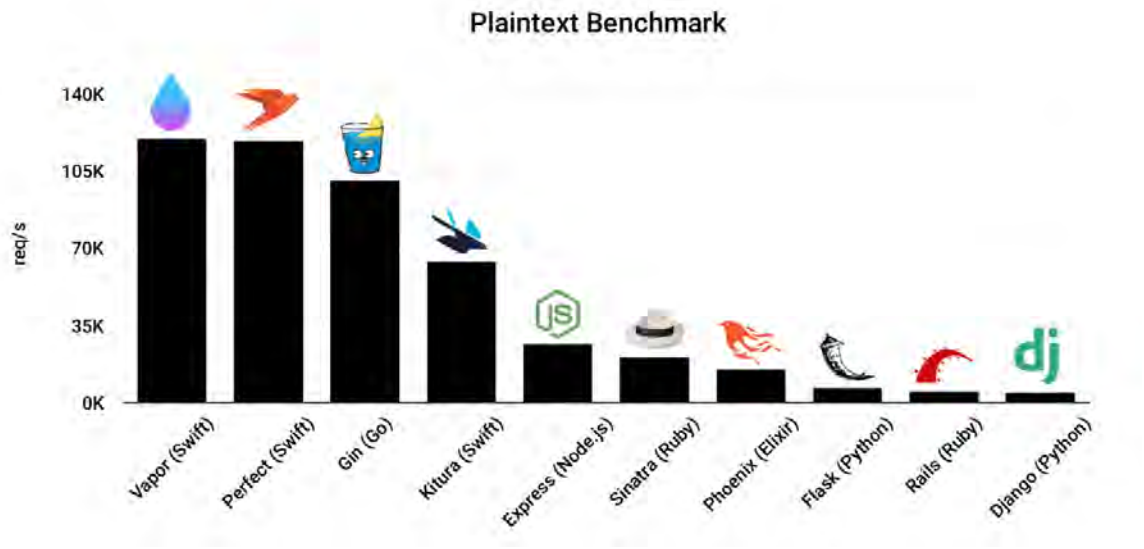
Framework Perfect rovnako poskytuje všetky služby a funkcie očakávané od moderného webového frameworku ako napríklad cookies, SSL, URL smerovanie, podporu JSON a iné. Za zmienku stojí podpora integrácie s širokým spektrom databázových systémov. Perfect je možné nasadiť ako samostatný webový server alebo ho spojiť s už existujúcou Apache alebo Nginx inštaláciou.

Oproti frameworku Kitura, Perfect ponúka asistenta pre konfiguráciu a nasadenie v podobe samostatnej aplikácie. V súčastnosti najnovšia verzia, perfect assistant 2.0, umožňuje jednoduchú nastavenie frameworku na základe šablón, správu závislostí, testovanie ale taktiež aj prepojenie s AWS[2] servermi, následné jednoduché nasadenie vytvorenej konfigurácie pre tieto serveri a v neposlednom rade aj integráciu, na platforme MacOS, s vývojovým prostredím Xcode od Apple.[20]

[2] Amazon Web Services

Vapor

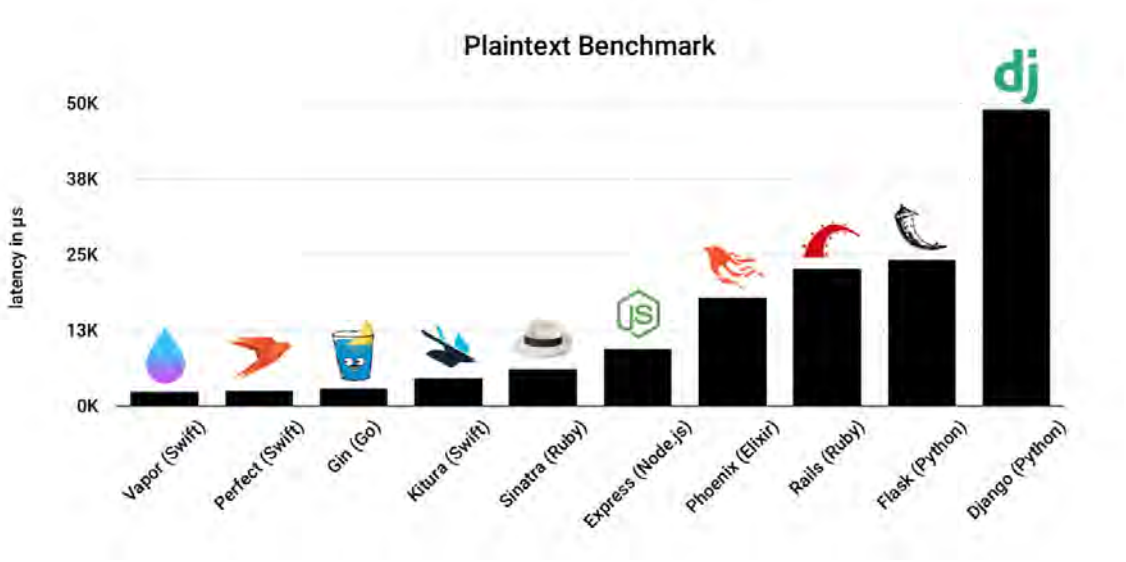
Vapor je ďalší z webových frameworkou dostupných pre programovací jazyk Swift. Jeho zdrojový kód je verejne dostupný na GitHub repozitári pod MIT licenciou. Prvá verzia, 0.1.0, bola vydaná, skupinou okolo programátora Tannera Nelsona, krátko po tom čo bol Applom Swift uvoľnený ako open-source koncom roku 2015. Verzie 1.0 a 2.0 nasledovali v rokoch 2016, respektíve 2017. Verzia 3.0, vydaná v roku 2018, priniesla kompletné prepracovanie frameworku čo značne zlepšilo jeho výkonnosť. Porovnanie výkonnosti webových frameworkou pre Swift a ostatné populárne programovacie jazyky je zobrazené na obrázkoch 3.1 a 3.2. Ako jeho základ bol použitý SwiftNIO čo je sieťový, neblokujúci framework pre Swift od Apple. Verzia 4.0 následne prišla až v roku 2020 a oproti verzií 3.0 priniesla len inkrementálne vylepšenia. Poslednou verziou v čase písania tejto kapitoly bakalárskej práce je verzia 4.40.0 z februára 2021.



Obr. 3.1: Výsledky z testu Plaintext benchmark. Webové frameworky sú v grafe zoradené podľa spracovaných žiadostí za sekundu. Väčšie hodnoty značia lepší výsledok.[24]

Tak ako frameworky Kitura 3.2 a Perfect 3.2 aj Vapor poskytuje služby a funkcionalitu očakávanú od moderného webového frameworku. Pre správu zdrojových kódov a potrebných balíčkov a súborov Vapor využíva štruktúru založenú na SPM[3]. To znamená rozdelenie na verejne dostupné súbory a dáta (cez FileMiddleware), zdrojové kódy, testy a dodatočne potrebné balíčky. V súčastnosti ekosystém okolo frameworku Vapor obsahuje cez 60 oficiálne podporovaných balíčkov ktoré umožňujú prácu s databázami, návrhovými jazykmi pre webové stránky a iných. Komunitou spravovaných balíčkov je pre Vapor framework v súčastnosti dostupných viac ako 120. V tejto sekcií sú priblížené tie z nich, ktoré sú podstatné v kontexte zamerania tejto bakalárskej práce. Konkrétne pôjde o ORM[4] framework Fluent a návrhový jazyk Leaf.[23]

[3]Swift Package Manager
[4]Object–relational mapping



Obr. 3.2: Webové frameworky zoradené podľa priemernej čakacej doby počas Plaintext benchmarku. Menšie hodnoty značia lepší výsledok.[24]

Fluent

Fluent je ORM framework pre Swift. Vyžíva silný typový systém Swiftu na vytvorenie ľahko použiteľného rozhrania pre prácu s databázou. Princípom práce s databázou cez Fluent je vytvorenie modelov, v programovacom jazyku Swift, ktoré odpovedajú dátovým štruktúram v databáze. Tieto modely sú následne používané na vytváranie či úpravu databáz, namiesto priamej interakcie s databázovým systémom. Fluent oficiálne podporuje PostgreSQL, SQLite, MySQL (z klasických SQL databáz) a MongoDB ako NoSQL databázu. Mimo oficiálne podporovaných databáz je ale možné získať podporu aj pre iné, a to vďaka podpore od komunity.

Pre reprezentáciu dát z databáz Fluent používa modely, v kontexte jazyka Swift reprezentované triedami ktoré obsahujú kódovateľné hodnoty a unikátne id. Pre definovanie kompletnej databázovej schémy je nutné použite migrácií ktoré definujú obmedzenia, indexy či cudzie kľúče ktoré sa v databáze vyskytujú.[25]

Leaf

Leaf je návrhový jazyk zo syntaxou inšpirovanou syntaxou programovacieho jazyka Swift. Jeho použitie v rámci frameworku Vapor je najčastejšie na generovanie dynamických webových stránok ale umožňuje vytvárať napríklad aj emaily.

Syntax Leafu je pomerne jednoduchá a pre Swift (ale aj iných) programátorov ľahko pochopiteľná. Nasleduje krátke zhrnutie možností syntaxe frameworku Leaf.

- token: # - signalizuje Leaf parseru začiatok tagu.
- meno tagu: alfanumerický reťazec.
- zoznam parametrov: **(param1, param2, ...)**.
- výrazy: **+, -, *, /, %, <, >, ==, || a iné**.

- riadiace príkazy: **#if, #else, #elseif**.
- cyklus: **#for, #endfor**.
- rozširovací tag: **#extend** - umožňuje rozširovať existujúce návrhy, typické použitie pre spoločné časti rôznych webových stránok.
- tagy pre import a export: **#import, #export** - používané pre uloženie respektíve načítanie uloženého obsahu.

Leaf dokáže pracovať s ľubovoľnými dátami pokiaľ sú konformné s Swift protokolom *Encodable*. Z toho vychádza nemožnosť pracovať priamo s poliami a nutnosť ich zabalenia v štruktúre.[25]

3.3 ARKit

Jedná sa o framework, ktorý bol vyvinutý spoločnosťou Apple a vydaný v roku 2017. Je určený výhradne pre zariadenia spoločnosti Apple, iPhone a iPad. Vytvorený bol s cieľom zjednodušiť prácu vývojárov pri tvorbe mobilných aplikácií pracujúcich s rozšírenou realitou. Prostredníctvom ARKitu dokáže aplikácia detektovať okolité prostredie a orientovať sa v ňom. ARKit taktiež umožňuje vkladanie virtuálnych objektov do reality a interakciu s nimi.

ARKit je podporovaný na operačných systémoch iOS od verzie 11 a po, v čase písania tejto kapitoly najnovšiu, verziu iOS 14. ARKit je rovnako podporovaný aj na operačnom systéme iPadOS, pre tablety, a to od jeho prvej verzie, iPadOS 13. Framework je podporovaný len na zariadeniach s SoC novšími ako A9. Niektoré funkcie frameworku vyžadujú vyšší model SoC ako A9. Napríklad pre detekciu a interakciu s osobami v AR scéne je potrebný A12 alebo novší. Rovnako, pre určité funkcie, môže byť vyžadovaná prítomnosť špeciálneho hardvéru, napríklad LiDAR[5] senzoru, na zariadení.[6]

Princíp práce ARKitu

Framework ARKit sa skladá z troch významných častí. Konkrétne sa jedná o časti pre sledovanie prostredia, porozumenie scéne a časti pre vykresľovanie výslednej scény a objektov v nej. Pre vykresľovanie virtuálnych objektov do reality ARKit používa knižnice SceneKit, SpriteKit a Metal.

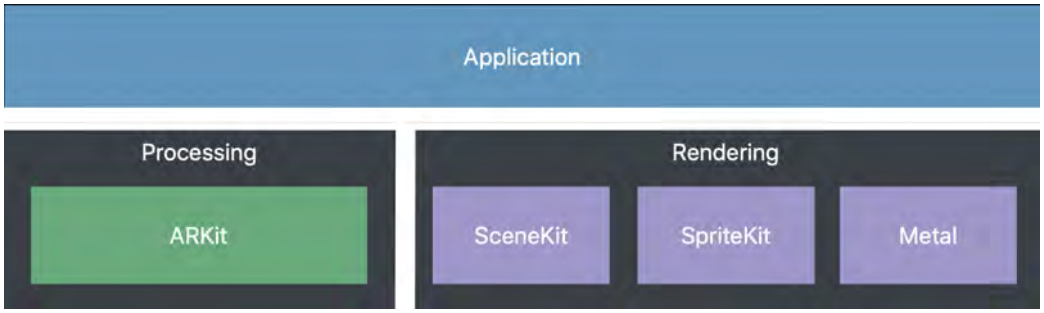
Metal slúži na prácu okolo vytvárania konkrétneho renderu. Knižnica SceneKit je používaná na vykreslenie 3D objektov a knižnica SpriteKit je použitá k manipulácii s 2D objektami. Všetky objekty ktoré sú zobrazované ARKitom v rámci scény sú spravované objektom `ARSession`.

Jednotlivé komponenty a ich závislosti bližšie zobrazuje obrázok 3.3. Objektu ARSession a jeho použitiu sa detailne venuje podsekcia 3.3.

Možnosti a konfigurácia ARKitu

ARKit ponúka široké možnosti práce s rozšírenou realitou. Ich konfiguráciu na základe konkrétnych požiadavkou danej aplikácie zabezpečuje trieda `ARConfiguration`. Logickú náväznosť tried ktoré ARKit pre svoje správne fungovanie a konfiguráciu využíva, popisuje

[5] laser imaging, detection and ranging

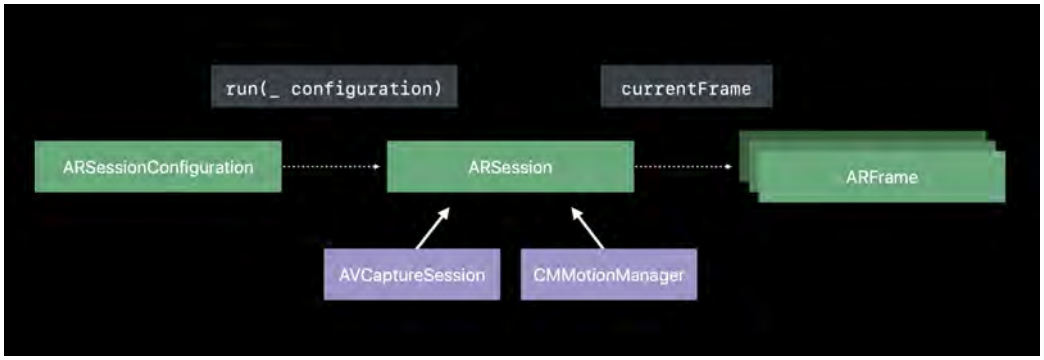


Obr. 3.3: Schéma komponentov frameworku ARKit ktoré sú využívané pre vytvorenie scény rozšírenej reality v mobilnej aplikácií. Správu AR sedenia má na starosti ARKit a vykresľovanie výslednej scény na obrazovke zariadenia je realizované za použitia SceneKit, Spritekit a Metal.[14]

obrázok 3.4. Pomocou podtried `ARSession` je možné ARKit využiť jedným z nasledovaných spôsobov: [3]

- **ARWorldTrackingConfiguration** - poskytuje sledovanie polohy a orientácie zariadenia vo vzťahu k akýmkoľvek povrchom, osobám alebo známym obrázkom a objektom, ktoré ARKit môže nájsť a sledovať pomocou zadnej kamery použitého zariadenia.
- **AROrientationTrackingConfiguration** - poskytuje základnú detekciu polohy zariadenia v skutočnom priestore.
- **ARGeoTrackingConfiguration** - používa GPS[6], elektronický kompas v zariadení a mapové dáta na sledovanie geografickej lokácie oblasti záujmu.
- **ARFaceTrackingConfiguration** - za pomoci prednej kamery používaného zariadenia detekuje tváre osôb nachádzajúcich sa v scéne, podporuje aj sledovanie výrazov tváre.
- **ARBodyTrackingConfiguration** - detekcia a sledovanie ľudského tela a jeho polohy v scéne, lietadiel a obrázkov ktoré sa nachádzajú v zábere zadnej kamery zariadenia.
- **ARImageTrackingConfiguration** - detekovanie určených obrazov pomocou zadnej kamery.
- **ARObjectScanningConfiguration** - zhromažďovanie vysoko presných priestorových údajov o okolí pomocou zadnej kamery zariadenia. Tieto informácie môžu byť použité k vytváraniu referenčných objektov pre využitie v aplikácií počas bežného používania.
- **ARPositionalTrackingConfiguration** - detekovnie polohy zariadenia v 3D prostredí.

[6]global positioning system



Obr. 3.4: Obrázok popisuje princíp náväzností jednotlivých tried framworku ARKit a teda jeho vnútorné fungovanie.[14]

Životný cyklus AR scény

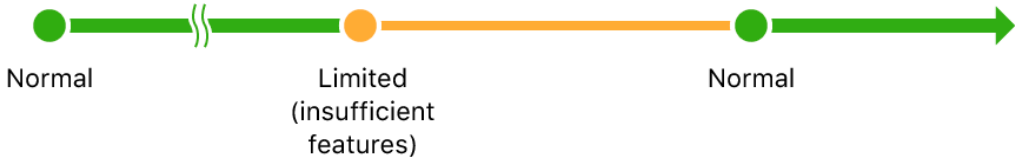
AR scény ktoré pracujú s reálnym svetom na jeho sledovanie využívajú techniku nazývanú visual-inertial odometry. Táto technika kombinuje dáta z pohybového senzoru zariadenia a počítačového videnia ktoré využíva kameru na zariadení. Tieto dáta sú skombinované a použité na sledovanie polohy a orientácie zariadenia v reálnom svete. Pre správnu funkcionalitu potrebuje sledovanie konzistentné dáta zo senzorov a obraz z kamery na ktorý je dostatočne rozmanitý alebo obsahuje rozpoznateľné objekty.

Po spustení ARKit potrebuje istý čas na zhromaždenie dostatku dát na určenie presnej polohy a orientácie zariadenia. ARKit umožňuje programátorovi zistiť stav v ktorom sa sledovanie okolia aktuálne nachádza cez metódy protokolu `ARSessionObserver` a vlastností triedy `ARCamera`. Proces inicializácie graficky znázorňuje obrázok 3.5. Po spustení `ARSession` je sledovanie nedostupné. Po spracovaní prvých dát s a dostáva do stavu limitované a akonáhle je ARKit schopný určiť polohu zariadenia s dostatočnou presnosťou, stav sa mení na normálny.[8]

Obr. 3.5: Obrázok zobrazuje stavy sedenia rozšírenej reality `ARSession` počas inicializácie.[8]

Počas používania frameworku ARKit aplikáciou sa presnosť sledovania okolia môže meniť. Toto je najčastejšie spôsobené akciami ktoré vykoná užívateľ alebo zmenou v reálnom prostredí. Pokiaľ dôjde k zníženiu presnosti sledovania pod prípustnú hranicu prepne sa stav sledovania do stavu limitované. V tomto stave sa neobnovujú a nedetekujú nové plochy a metódy ARKitu pracujúce s reálnym prostredím nevracajú výsledky. Táto situácia je bližšie znázornená na obrázku 3.6.

Pri používaní môže nastať aj situácia kedy ARKit nieje schopný pokračovať v sledovaní reálneho svetu a dochádza k prerušeniu sledovania. Typickým dôvodom pre tento stav je prerušenie aplikácie alebo rapídna zmena reálneho prostredia. ARKit sa po prerušení môže pokúsiť o obnovu sledovania. Toto správanie je možné zmeniť k tomu určenou metódou `sessionShouldAttemptRelocalization()`. Pre programátora je dôležité dať užívateľovi



Obr. 3.6: Obrázok zobrazuje možné zmeny sledovania okolia ARKitom ktoré môžu nastať počas používania. Zmeny môže spôsobiť užívateľ interakciou alebo môžu byť následkom zmeny prostredia.[8]

možnosť zahodiť scénu a začať ju vytvárať odznova v prípade, že by ARKit nebol schopný sledovanie pôvodnej scény obnoviť. Proces stavu sledovania pri prerušení popisuje obrázok 3.7.

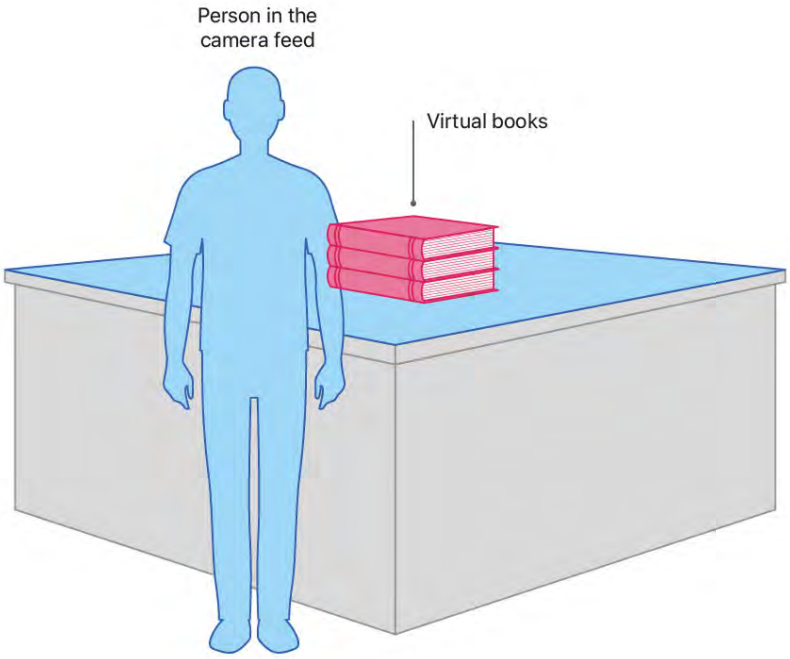
Obr. 3.7: Obrázok zobrazuje stavy AR sedenia pred, počas a po prerušení. Prerušenie AR sedenia môže nastať na základe opustenia aplikácie, zmenou prostredia či zlyhania sledovacej schopnosti ARKitu.[8]

Sledovanie ľudského tela

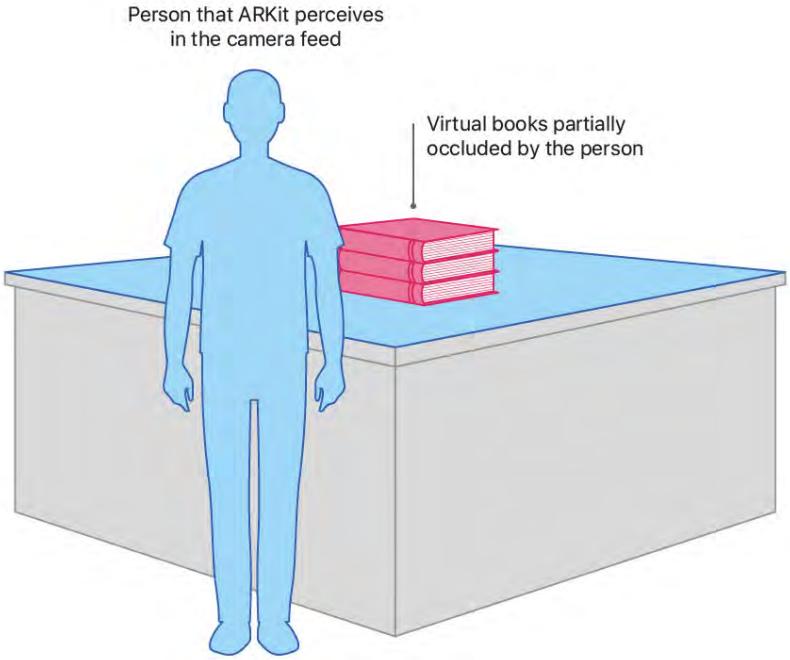
ARKit umožňuje sledovanie ľudského tela a jeho polohy v AR scéne dvoma rôznymi spôsobmi. Na základe konfigurácie buď deteguje pozíciu ľudského tela v scéne za účelom lepšej interakcie s virtuálnymi objektami alebo sleduje jeho presnú pozíciu a pohyby v 3D priestore za účelom ďalšieho spracovania. Pre použitie tejto funkcionality je vyžadované zariadenie s SoC A12 alebo novším. Pre sledovanie ľudského tela v scéne je potrebné ARKit konfigurovať pomocou inštancie triedy `ARBodyTrackingConfiguration`. Pokiaľ ARKit v scéne deteguje ľudské telo, vracia objekt `ARBodyAnchor` ktorý obsahuje pozíciu detekovaného tela v 3D priestore, konkrétne sa jedná o pozíciu bedrového kĺbu.

Pre správne vykresľovanie virtuálnych objektov vzhľadom k osobám v scéne už žiadna ďalšia konfigurácia nieje potrebná. ARKit framework na základe pozície osoby a virtuálneho predmetu zaistí aby bol objekt vykreslený správne. Rozdiel medzi AR scénou bez aktivovanej detekcie ľudského tela a s jej aktívnym použitím detailnejšie znázorňujú obrázky 3.8 a 3.9.

Detailnejšie sledovanie pozície ľudského tela a končatín ARKit umožňuje pomocou poskytnutia pozícií jednotlivých kĺbov detekovaného ľudského tela a triedu `ARSkeleton` ktorá slúži ako rozhranie k sledovaným kĺbom. V závislosti na požiadavkoch na sledovanie je možné použiť triedu `ARSkeleton2D` pre sledovanie polohy v 2D priestore alebo `ARSkeleton3D` pre sledovanie polohy tela v 3D priestore. Pre transformáciu detekovanej polohy kĺbu na model tela umiestnený v ARscéne je možné využiť metódu `modelTransform()` ktorá vracia relatívnu polohu daného kĺbu voči bedrovému kĺbu.[6]



Obr. 3.8: Obrázok zobrazuje postavu v AR scéne bez použitia funkcionality ARKitu ktorá umožňuje detekciu polohy ľudského tela v AR scéne. Virtuálne objekty sú v tomto prípade vykresľované pred postavu aj keď sa v scéne nachádzajú za ňou. [19]



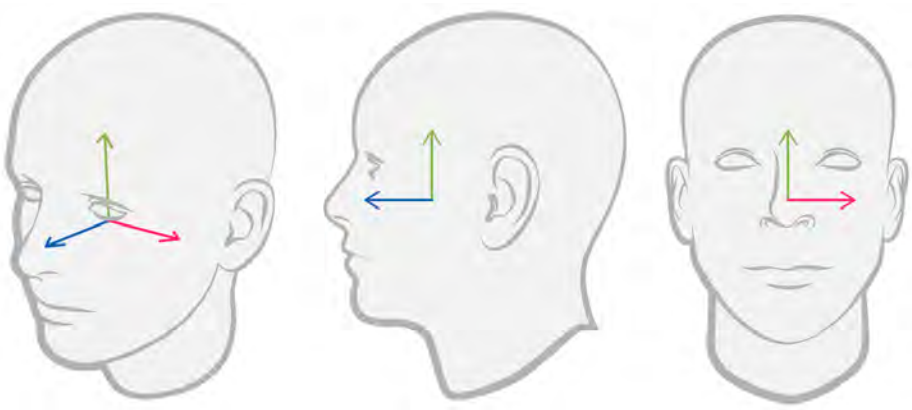
Obr. 3.9: Obrázok zobrazuje postavu v AR scéne s použitím funkcionality ARkitu ktorá umožňuje detekciu polohy ľudského tela v AR scéne. Virtuálne objekty sú v tomto prípade vykresľované realisticky vzhľadom na polohu postavy.[19]

Sledovanie tváre a mimiky

Pomocou frameworku ARKit je možné v mobilných aplikáciách detektovať tvár a sledovať jej polohu, orientáciu či mimiku. Sledovanie tváre je konfigurovateľné pomocou inštancie triedy `ARFaceTrackingConfiguration`. Táto funkcionalita však vyžaduje kompatibilné zariadenie, konkrétne je nutná prítomnosť takzvanej TrueDepth kamery. Jedná sa o prednú kameru na zariadení ktorá je schopná detegovať tvár, jej črty a výraz v 3D prostredí. TrueDepth kameru môžeme nájsť na všetkých zariadeniach ktoré disponujú technológiou FaceID (iPhone X a novšie, s výnimkou iPhone SE 2). Od verzie 4, ARKit na vybraných zariadeniach s SoC A12 Bionic podporuje súčasné použitie detekcie tváre pomocou prednej kamery a detekciu okolitého prostredia pomocou zadnej kamery.

Pri detekcií tváre ARKit pridáva do AR sedenia objekty `ARFaceAnchor`. `ARFaceAnchor` obsahuje informácie o polohe a orientácií detekovanej tváre. Pre udanie pozície a polohy tváre v AR scéne ARKit používa pravostranný systém súradníc, kde pozitívny smer na osi x smeruje k ľavej strane (z pohľadu detekovanej tváre), pozitívny smer na osi y smeruje nahor (voči tvári, nie AR scéne) a pozitívny smer na osi z smeruje k pozorovateľovi (užívateľovi). Detaily sú znázornené na obrázku 3.10.

Pre prácu s detekovanou tvárou používateľa (nanesenie textúry, pridávanie virtuálnych objektov) poskytuje ARKit objekt `ARFaceGeometry` ktorý obsahuje detailnú topológiu (rozmer a tvar) detekovanej tváre. Taktiež sú k dispozícií aj informácie o aktuálnom výraze sledovanej tváre.



Obr. 3.10: Obrázok zobrazuje detekovanú tvár a súradnicový systém pre jej polohu a orientáciu.[6]

LiDAR a Depth API

S príchodom zariadení obsahujúcich LiDAR senzor (iPad Pro 11 2. generácie, iPad Pro 12.9 4. generácie, iPhone 12 Pro a iPhone 12 Pro Max) uverejnil Apple aj Depth API. Toto API, v kombinácií s klasickou detekciou scény cez fotoaparát zariadenia, umožňuje získať informácie o 3D prostredí omnoho rýchlejšie. S pohľadu užívateľa je možné dosiahnuť takmer okamžitú detekciu prostredia a následné umiestnenie virtuálneho objektu.

Depth API neumožňuje len rýchlejšiu detekciu 3D prostredia ale prináša aj jej vyššiu presnosť, v ideálnych podmienkach až na úroveň jednotlivých pixlov. To umožňuje vytvoriť ešte reálnejšie pôsobiace scény a značne vylepšiť interakciu virtuálnych objektov či už s prostredím alebo osobami ktoré sa v scéne nachádzajú.

Kapitola 4

Návrh systému

Cieľom práce je vytvorenie mobilnej aplikácie pre mobilný operačný systém iOS a webového serveru s ktorým bude aplikácia spolupracovať. Samotná aplikácia bude umožňovať vizualizáciu obrazov na vertikálnych plochách prostredníctvom rozšírenej reality. Server bude poskytovať databázu obrazov a umožňovať ich spravovanie.

Aplikácia je určená pre mobilný operačný systém od Apple Inc. (iOS) a tomu zodpovedá výber použitých technológií. Pre prácu s rozšírenou realitou to je framework ARKit a pre vytvorenie užívateľského rozhrania používa UIKit.

Webový server je určený pre ľubovolný MacOS respektíve Linux systém a bude implementovaný pomocou programovacieho jazyka Swift a jeho frameworkou. V tejto kapitole sa venujem špecifikácii požiadavkou na funkcionalitu aplikácie a serveru a následne návrhu spôsobu ich realizácie.

4.1 Určenie systému klient-server

Systém pozostávajúci z aplikácie a webového serveru má vo svojej podstate potencionálne širokú škálu využití. Či už sa jedná o použitie bežnými používateľmi ktorý majú záujem vidieť, u seba doma na stene, obrazy ktoré nevlastnia ale o ich zaobstaraní uvažujú alebo zákazníkov galérií, aukčných siení či spoločností ktoré poskytujú tlač fotografií. Rovnako do úvahy prichádza poskytnutie a upravenie systému priamo pre potreby konkrétnych galérií či aukčných siení.

Cieľová skupina

Výsledná aplikácia je cielená na bežných užívateľov mobilných zariadení ktorý chcú vidieť daný obraz prípadne fotografiu na svojom mieste na stene ešte pred tým než sa dostanú k fyzickému exempláru. Taktiež sa môže jednať aj o už spomínaných zákazníkov galérií a iných spoločností. Očakávaná cieľová veková skupina je preto staršia ako 15 rokov a nepredpokladá sa, že užívatelia majú akúkoľvek predchádzajúcu skúsenosť s rozšírenou realitou.

Webový server je navrhovaný pre použitie správcami či kurátormi daných spoločností, galérií či aukčných siení ktoré systém využívajú. Predpokladaná cieľová skupina pre serverovú časť je preto staršia a predpokladá sa základná skúsenosť s organizovaním a správou obrazov v elektronických systémoch.

Podporované zariadenia

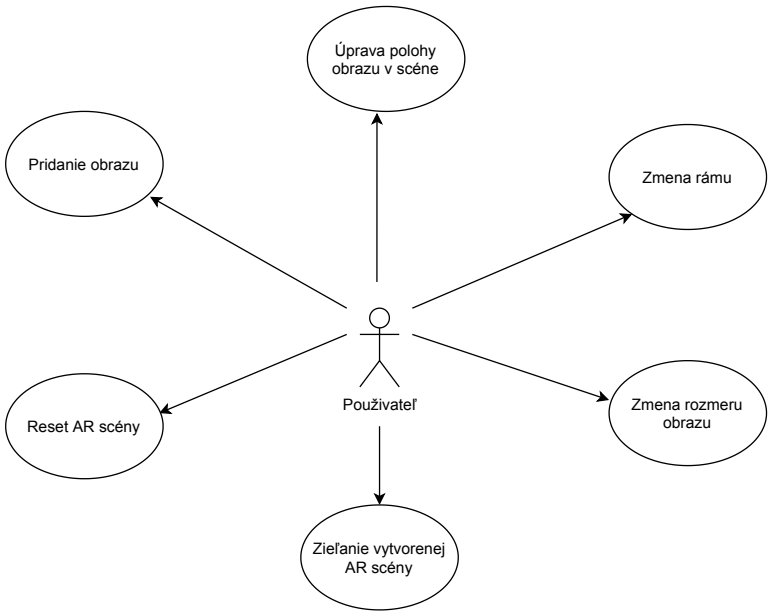
Aplikácia je navrhnutá pre zariadenia s mobilným operačným systémom iOS. Jedná sa o smartfóny iPhone ktoré obsahujú minimálne SoC A9 alebo novší. Tieto zariadenia disponujú veľkosťou displeja v rozmedzí od 4,7 palca do 6,5 palca. Táto skutočnosť bola preto zohľadnená pri návrhu grafického užívateľského rozhrania aplikácie.

Serverová implementácia je navrhnutá tak aby podporovala najčastejšie používané operačné systémy v tejto oblasti. Podporované sú všetky operačné systémy s Linuxovým jadrom a rovnako je podporované spustenie na operačnom systéme MacOS.

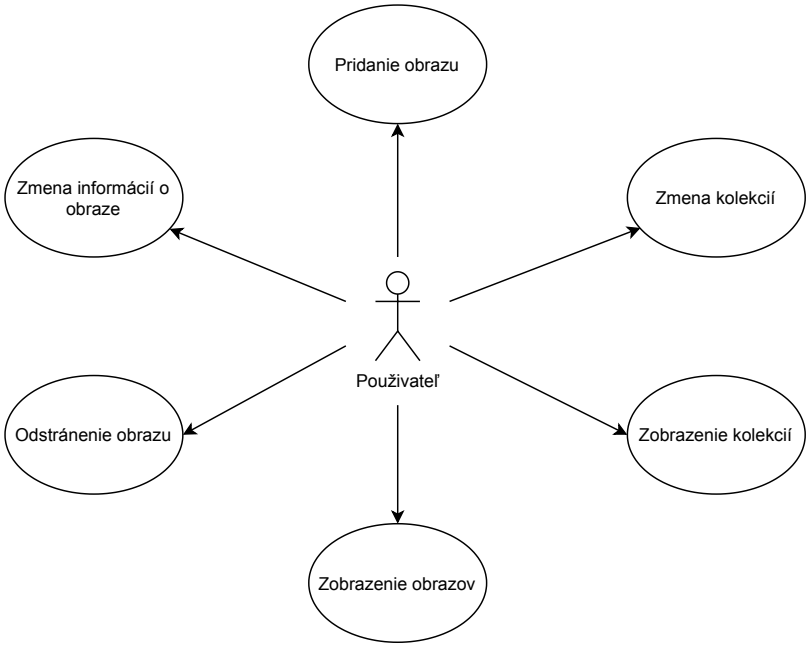
4.2 Požadovaná funkcionalita

Hlavnou úlohou aplikácie je prezentovať vybrané obrazy čo najvernejšie t.j. docieliť pri zobrazení v rozšírenej realite reálne rozmery, presné a spoľahlivé umiestnenie obrazu v rámci AR scény, reálnu textúru prípadne odlesky. Medzi ďalšie požiadavky patrí interakcia s objektami, zmena ich vlastností (veľkosť) či možnosť výslednú scénu zachytiť a zdieľať. Možné použitia aplikácie znázorňuje aj diagram prípadov použitia ktorý je zachytený na obrázku 4.1. Jedná sa o štandardný diagram prípadov použitia v jazyku UML.

Pre server sú hlavné požiadavky nasledovné. V prvom rade, uchovávanie obrazov a informácií k nim a následne sprístupnenie týchto informácií pre mobilnú aplikáciu. Taktiež je na mieste požiadavka na možnosť zatriedenia uložených obrazov do kolekcií a prípadné upravovanie informácií k nim pripojených. UML diagram na obrázku 4.2 znázorňuje všetky uvažované použitia serveru.



Obr. 4.1: Diagram prípadov použitia zobrazujúci jednotlivé možnosti použitia vyvíjanej mobilnej aplikácie užívateľom.



Obr. 4.2: Diagram prípadov použitia zobrazujúci jednotlivé možnosti použitia vyvíjaného webového serveru jeho užívateľom.

Vkladanie obrazov na vertikálne plochy

Vkladanie obrazov na vertikálne roviny je hlavná funkcionalita navrhovanej mobilnej aplikácie. Pre jej realizáciu potrebné detekovať vertikálne plochy v reálnom prostredí okolo užívateľa za pomoci frameworku ARKit a následne na ne umiestniť vybrané obrazy. Pre dosiahnutie čo najvernejšej podoby virtuálnej reality a skutočnosti bude potrebné zachovanie pôvodných rozmerov obrazu, verné reprodukcie farieb či textúry daného obrazu. Dôležité budú aj odlesky materiálu ktoré by mali zodpovedať reálnemu osvetleniu v prostredí.

Úprava vlastností obrazu

Pre vytvorenie scény v rozšírenej realite "na mieru" pre každého užívateľa je potrebné umožniť istú mieru prispôsobenia obrazov. Najjednoduchšou a najprirodzenejšou možnosťou je výber osobitého rámu pre daný obraz. Medzi ďalšie možnosti môžme zaradiť zmenu rozmerov prezentovaného obrazu. Pre dobrý užívateľský zážitok je žiadúce aby zmeny bolo možné aplikovať aj po umiestnení obrazu na vertikálnu plochu do scény v rozšírenej realite.

Úprava pozície obrazu

Úprava pozície obrazu je vedľajšia no podstatná funkcia pre pohodlnejšiu prácu s aplikáciou. Po umiestnení obrazu na detekovanú plochu by mal byť užívateľ schopný s obrazom manipulovať a teda ho napríklad presunúť, prípadne obraz zo scény aj odstrániť. Presun obrazu je logicky možný len v rámci detekovanej vertikálnej plochy a preto by mal užívateľ byť informovaný o jej hraniciach. V úvahu pripadajú viaceré riešenia. Jedným z nich je nerušivo užívateľovi zobraziť hranice detekovanej plochy priamo v AR scéne. Druhá možnosť je upozornenie užívateľa pokiaľ by sa obraz pokúsil presunúť mimo detekovanú plochu,

napríklad haptickou odozvou, znemožnením presunu alebo textovým upozornením na prekročenie hraníc plochy, na ktorú je možné obraz spoľahlivo umiestniť, v aplikácií.

Zdieľanie výslednej scény

Pri vytvorení scény s ktorou je užívateľ spokojný je na mieste požiadavka na jej uloženie či zdieľanie. Ideálnym formátom je v tomto prípade zhotovenie snímky obrazovky ktorá bude zachytávať, užívateľom vytvorenú, scénu v rozšírenej realite. Zdieľanie by malo byť v aplikácii na zariadení umožnené buď lokálne (zdieľanie do inej aplikácie na zariadení či jednoduché uloženie) prípadne vzdialene pomocou správy, emailu či odoslaním pomocou inej komunikačnej aplikácie ktorá je na zariadení dostupná.

Správa obrazov

Podstatnou funkciou webového serveru je možnosť ukladať respektíve meniť obrazy a informácie o nich. Pre podporu viacerých galérií a sprehľadnenie obrazov v nich bude vhodné obrazy členiť na kolekcie ktoré budú prislúchať jednotlivým galériám. Okrem samotných obrazov je pre správne fungovanie mobilnej aplikácie potrebné uchovávať aj informácie o materiáloch z ktorých sú zhotovené, ich reálnych rozmerov, cene, o ich pôvode, autorovi a taktiež aj krátkeho popisu daného obrazu. Štruktúry navrhnuté pre uchovávanie potrebných dát nájdete na obrázku 4.3.

Webové API

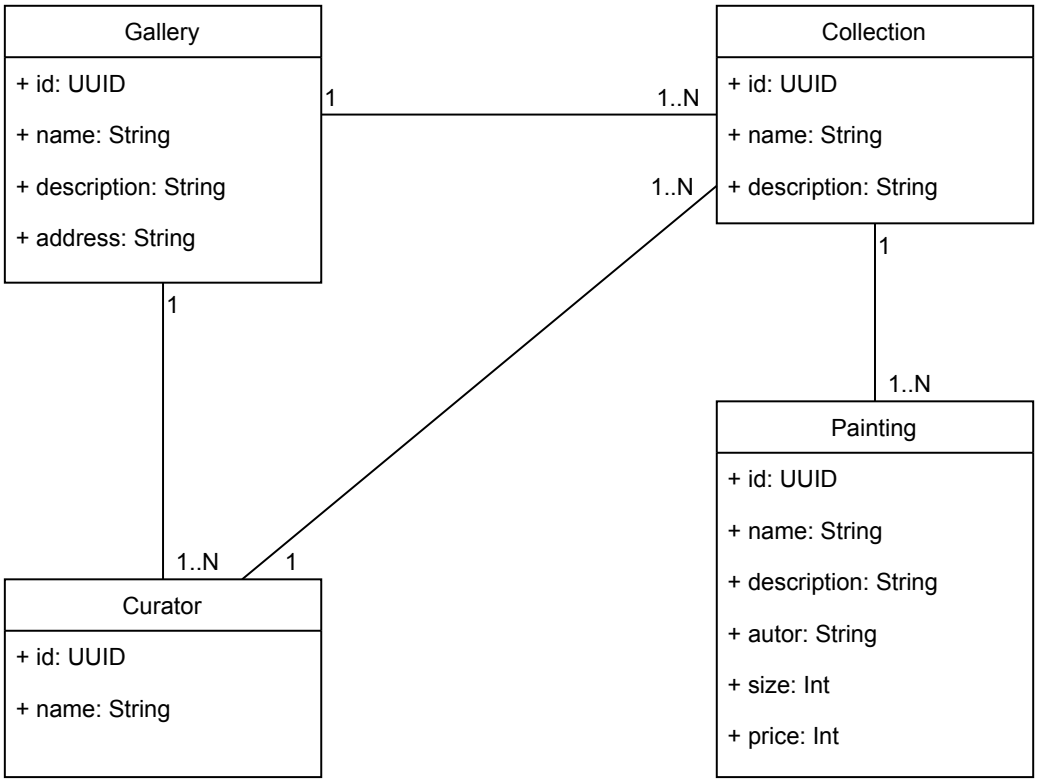
Pre komunikáciu mobilnej aplikácie s databázou na webovom serveri je potrebné na webovom serveri implementovať API, cez ktoré budú obrazy a informácie o nich sprístupnené. Za predpokladu webového spojenia zo serverom, bude cez toho API mobilná aplikácia získať všetky dostupné obrazy a informácie o nich pre danú galériu či kolekciu, eventuálne si tieto informácie aktualizovať.

4.3 Návrh UI/UX

Na základe požiadaviek, na funkcionalitu mobilnej aplikácie a webového serveru, zadefinovaných v predchádzajúcej sekcií, zo zameraním sa na vybranú cieľovú skupinou užívateľov a s zohľadnením odporúčaní pre návrh aplikácií pracujúcich s rozšírenou realitou na platforme iOS bolo navrhnuté používateľské rozhranie určené pre výslednú aplikáciu a webový server.

Použité nástroje

Pri tvorbe návrhu som sa rozhodol využiť program určený k tomuto cieľu a to, XD od Adobe. Tento nástroj mi umožnil navrhnúť jednotlivé obrazovky aplikácie, prepojiť ich do wireframu a ten následne pomocou mobilnej aplikácie na iPhone vyskúšať na reálnom zariadení bez nutnosti každú zmenu v návrhu implementovať. Táto funkcionalita mi pomohla prvky navrhnúť tak aby na zariadení pôsobili zamýšľaným dojmom a ušetrila nemalý čas pri procese návrhu a testovania.



Obr. 4.3: Počiatočný návrh UML diagramu štruktúr pre uloženie dát v databáze a prácu s nimi. Štruktúra `Painting` reprezentuje konkrétny obraz, štruktúra `Gallery` slúži na uchovanie údajov o galérií, štruktúra `Curator` uchováva dáta o kurátorovi a štruktúra `Collection` reprezentuje dáta o kolekcií danej galérie. Väzby medzi jednotlivými štruktúrami, a ich kardinalita vychádzajú z logických požiadavkou na funkcionalitu systému.

Proces návrhu

Samotný proces návrhu užívateľského rozhrania nebol jedno krokový ale jednalo sa o iteratívny proces. Po navrhnutí prvej verzie užívateľského rozhrania a jeho vyskúšaní pomocou mobilnej aplikácie Adobe XD (v prípade rozhrania mobilnej aplikácie) alebo v prehliadači (pri webovom serveri) boli do návrhu zapracované zmeny vychádzajúce z tohto testovania. Jednalo sa najmä o zmenu veľkosti a umiestnenia prvkov užívateľského rozhrania.

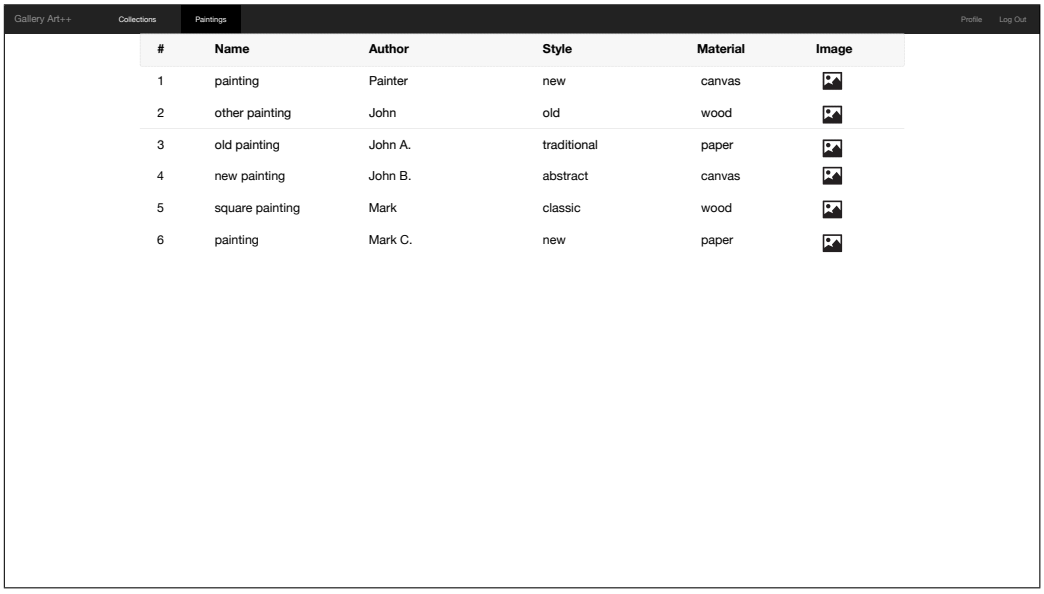
V prípade mobilnej aplikácie ďalšie iterácie užívateľského rozhrania vznikali po implementovaný prvého prototypu aplikácie a získaní spätnej väzby od užívateľov ktorý mali možnosť túto verziu aplikácie testovať. Spôsobu testovania a získaným výsledkom sa bližšie venujem v kapitole 6. Bližšie informácie o jednotlivých iteráciách návrhu sa nachádzajú v sekcií 4.3.

Prvotný návrh

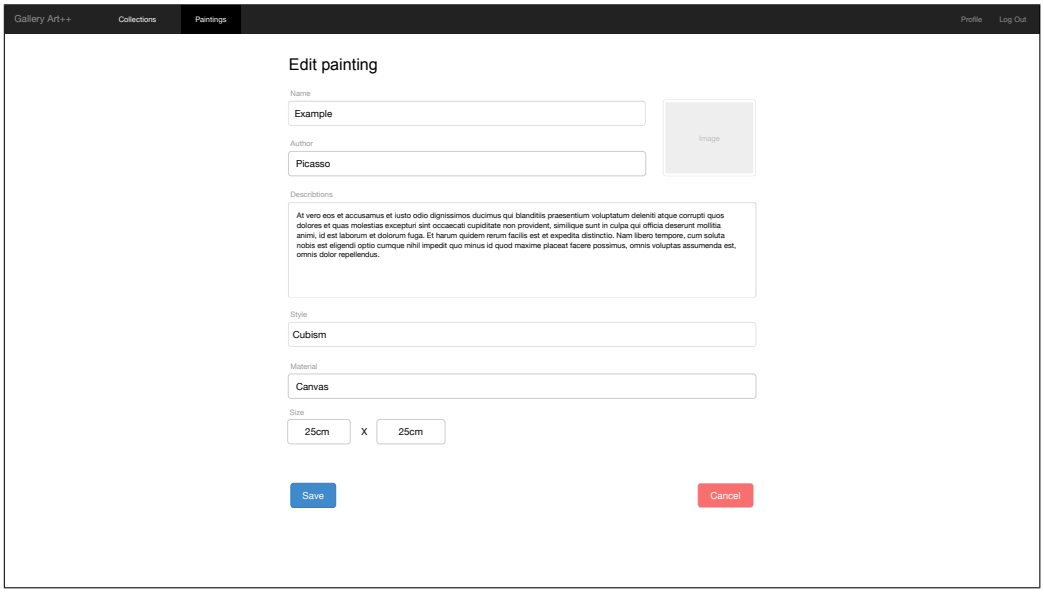
Prvotný návrh bol vypracovaný v aplikácií Adobe XD za účelom otestovania použiteľnosti a získania spätnej väzby od užívateľov. Do aplikácie bola implementovaná až nasledujúca verzia v ktorej boli odstránené najväžnejšie nedostatky. Podrobnejšie údaje možno nájsť v kapitole 6. Za zmienku stojí odstránenie úvodnej obrazovky ktorá plnila úlohu akéhosi medzi-kroku pred spustením samotnej časti aplikácie ktorá využívala rozšírenú realitu. Niž-

šie sa nachádza obrázok 4.6 ktorý znázorňuje a popisuje hlavné obrazovky tohto prvotného návrhu užívateľského rozhrania aplikácie.

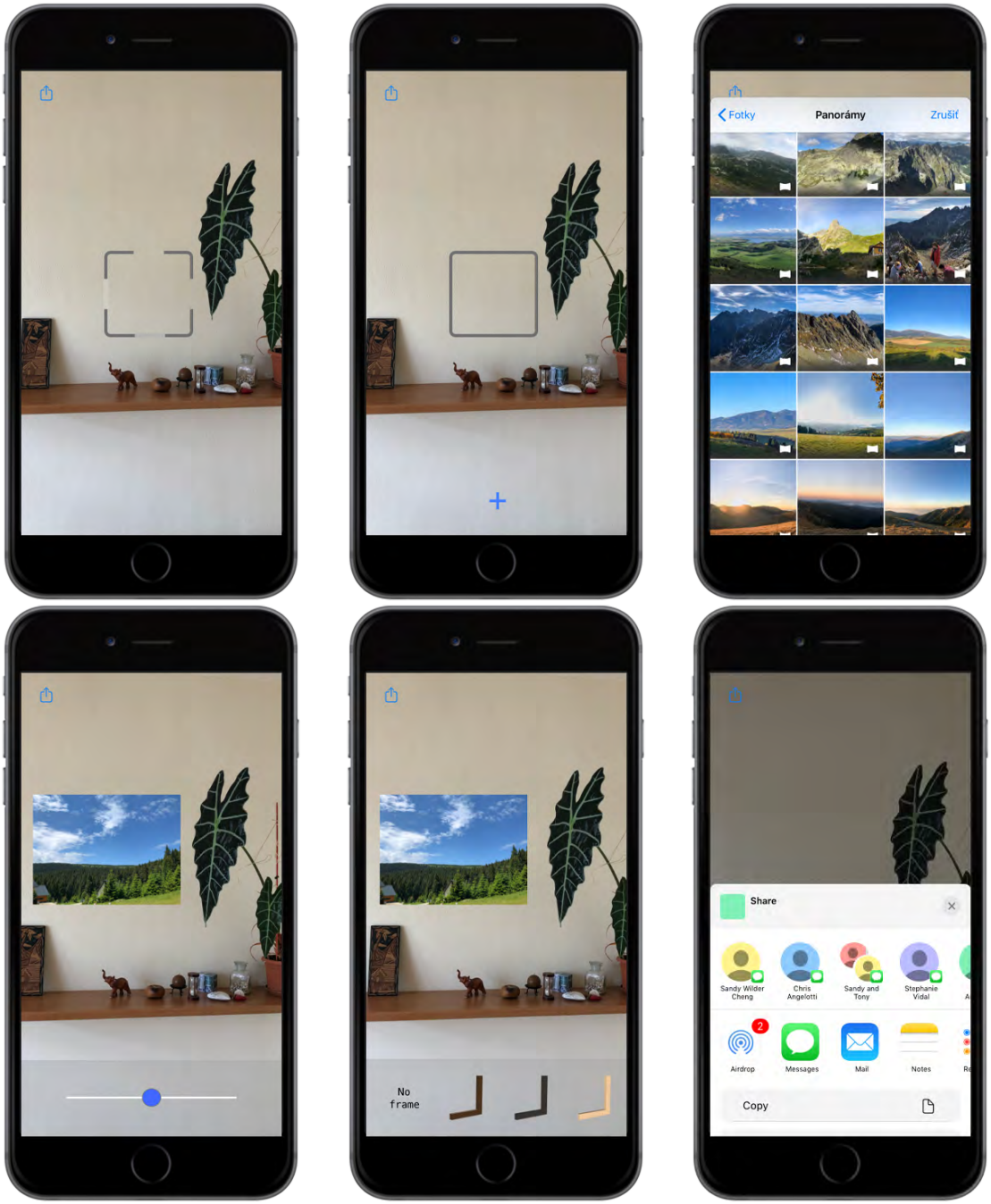
Návrh užívateľského rozhrania pre webový server bol rovnako navrhnutý v aplikácií Adobe XD a následne priamo implementovaný. Zmeny tohto pôvodného návrhu a ich odôvodnenie sú popísané v kapitole 6. Vybrané časti užívateľského rozhrania webového serveru pre správu obrazov znázorňujú obrázky 4.4 a 4.5.



Obr. 4.4: Ukážka užívateľského rozhrania webového serveru pre správu obrazov. Jedná sa o zoznam obrazov a k nim prislúchajúcich najdôležitejších informácií.



Obr. 4.5: Ukážka užívateľského rozhrania webového serveru pre správu obrazov zobrazujúca detailné informácie o konkrétnom obraze (názov obrazu, meno autora, popis obrazu, umelecký štýl, materiál obrazu, fyzické rozmery obrazu a vizuálna podoba obrazu).

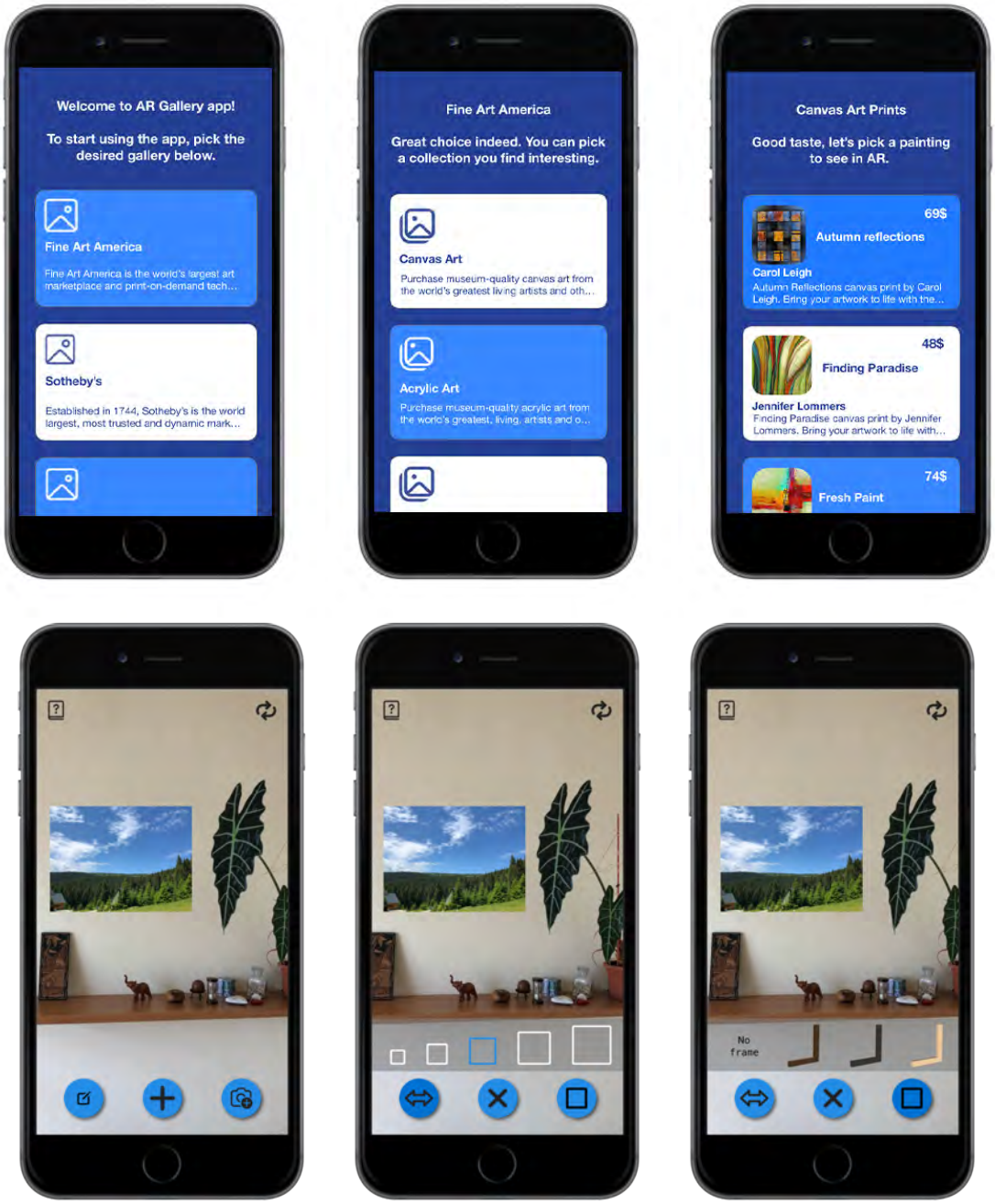


Obr. 4.6: Ukážka prototypu používateľského rozhrania aplikácie. Prvý rad zľava: inicializácia AR scény, úspešné detekovanie plochy na umiestnenie objektu a výber objektu na pridanie do scény. Druhý rad zľava: vybraný obraz umiestnený v AR scéne s posuvníkom na zmenu veľkosti obrazu, výber rámu pre obraz, výber spôsobu zdieľania scény. Na obrázku sú znázornené sú len vybrané hlavné obrazovky aplikácie.

Finálny návrh

Finálny návrh užívateľského rozhrania aplikácie vznikol po iteratívom procese návrhu ktorý bol založení na užívateľskom testovaní respektíve spätnej väzbe získanej počas tohto testovania. Celkovo návrh pred svojou finálnou podobou prešiel troma iteráciami. Finálnu podobu návrhu zobrazuje a popisuje obrázok 4.7. Detailnejšie informácie o spätnej väzbe od užívateľov a z nej vychádzajúcich zmien nájdete v kapitole 6 venovanej testovaniu.

Pri užívateľskom rozhraní webového serveru pre správu obrazov bol rovnako použitý iteratívny proces návrhu, ktorý čerpal zo spätnej väzby získanej počas testovania od používateľov. Narozdiel od mobilnej aplikácie tento návrh nepotreboval tak výrazné zmeny. Detailom o zmenených prvkoch užívateľského rozhrania a dôvodom k tomu vedúcim sa venujem v kapitole 6.



Obr. 4.7: Ukážka návrhu výsledného používateľského rozhrania aplikácie. Prvý rad zľava: úvodná obrazovka a výber galérie, výber kolekcie na základe zvolenej galérie a výber obrazu zo zvolenej kolekcie pre pridanie do scény. Druhý rad zľava: vybraný obraz umiestnený v AR scéne, zmena veľkosti obrazu, výber rámu pre obraz. Znázornené sú len vybrané hlavné obrazovky aplikácie.

Kapitola 5

Implementácia

Implementácia mobilnej aplikácie a webového serveru vychádzala z funkčných požiadavkou bližšie špecifikovaných v sekcií 4.2. Grafické užívateľské rozhranie bolo implementované podľa návrhu ktorý sa nachádza v kapitole 4.3. Použitý programovací jazyk pre implementáciu je Swift, vývoj prebiehal v IDE[1] Xcode od Apple Inc. Mobilná aplikácia je implementovaná pre zariadenia iPhone ktoré podporujú ARKit, vid. sekcia 3.3, a webový server je možné spustiť na platformách MacOS a Linux. V kapitole je najskôr popísaná implementácia webového serveru, databáze, API a užívateľského rozhrania respektíve webovej stránky. Druhá časť kapitoly popisuje spôsob implementácie mobilnej aplikácie.

5.1 Webový server

Táto sekcia popisuje implementáciu webového serveru, ktorá bola realizovaná pomocou frameworku Vapor v programovacom jazyku Swift. V tejto kapitole čitateľ nájde popis implementácie pre databázový systém, založený na PostgreSQL pre správu obrazov a ostatných údajov potrebných pre plnenie cieľov systému, webové API na sprístupnenie týchto dát mobilnej aplikácií a webové rozhranie, respektíve webovej stránky, pomocou ktorého je možné pridávať a spravovať dáta v systéme. Vzhľadom na použitie programovacieho jazyka Swift, jeho frameworkou a charakteru systému vychádza výsledná implementácia webového serveru z architektúry `Model-View-Controller`.

Databáza

Databázový systém použitý pre implementáciu v rámci webového serveru je PostgreSQL. Jedná sa o jeden z mnohých rozšírených systémov pre správu databáz. Zvolený bol pre kompatibilitu s ostatnými použitými technológiami, najmä frameworkom Vapor, a širokú dostupnosť dokumentácie a podpory. Pre prácu s databázovým systémom v rámci serveru bol využitý ORM framework Fluent.

Fluent pre prácu s databázovým systémom využíva modelové typy ktoré reprezentujú dátové štruktúry v databáze. Databázové operácie sú následne realizované prostredníctvom týchto modelov namiesto nutnosti použitia priamych SQL dotazov. Jednotlivé štruktúry z návrhu databáze boli pri implementácií prevedené na modely v jazyku Swift. Ich implementácia sa nachádza v priečinku `Models` v súboroch `Collection.swift`, `Gallery.swift`,

[1] integrated development enviroment

`Painting.swift` a `Curator.swift`. Pre vytvorenie databázovej štruktúry pri prvom spustení systému boli implementované prislúchajúce migrácie.

API

Komunikácia databázovej časti serveru a mobilnej aplikácie je zabezpečená cez webové REST API. Údaje o obrazoch, kolekciách a galériách sú sprístupnené cez koncové body ktoré poskytujú informácie vo formáte `JSON` cez `HTML` dotazy typu `GET` a `POST`. Implementácia API sa nachádza v súbore `APIController.swift`. Po prijatí a dekódovaní požiadavky na konkrétne dáta, webový server vykoná daný dotaz na databázový systém, cez vyššie spomínaný model, a získané dáta odošle ako odpoveď. API umožňuje prístup k všetkým údajom ktoré sa nachádzajú v databázovom systéme pomocou nasledovných koncových bodov.

- `/api/galleries` - Sprístupnenie dát o existujúcich galériách.
- `/api/collections` - Sprístupnenie dát o existujúcich galériách.
- `/api/paintings` - Sprístupnenie dát o všetkých obrazoch v systéme.
- `/api/paintingsForCollection/:collectionID` - Sprístupnenie dát o všetkých obrazoch pre danú kolekciu.
- `/api/collectionsForGallery/:galleryID` - Sprístupnenie dát o kolekciách pre danú galériu.

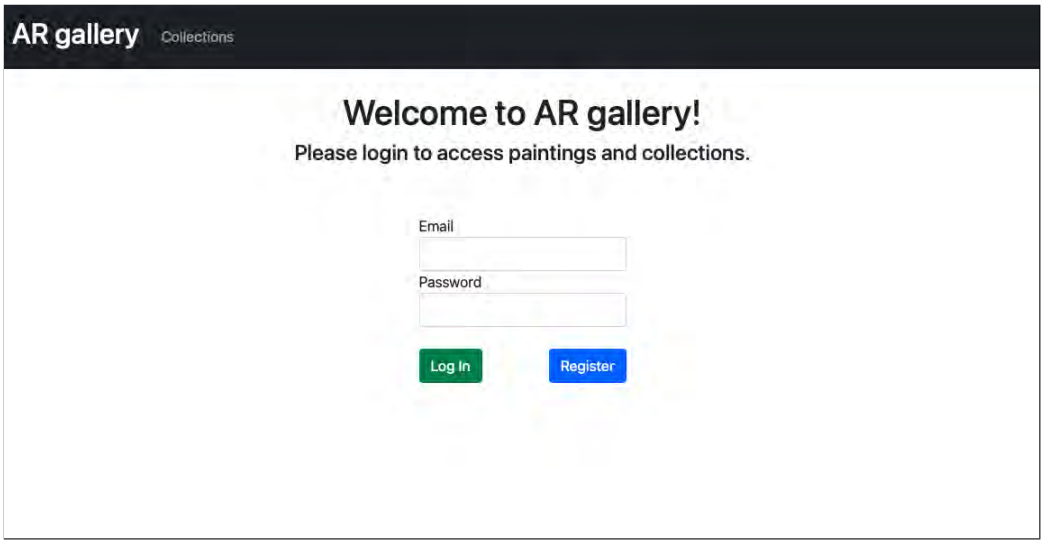
Samotné obrazy sú v rámci serveru uložené na verejne dostupnom priečinku. Cesta k jednotlivým obrazom sa nachádza v databázovom systéme a je súčasťou dát dostupných cez API. O sprístupnenie týchto dát sa stará `FileMiddleware()` ktorý sprostredkuje prístup k verejne dostupnému priečinku na serveri.

Webové užívateľské rozhranie

Pre správu a pridávanie obrazov a kolekcií bolo implementované webové rozhranie za použitia vzorovacieho jazyku Leaf a Swiftového frameworku Vapor a frameworku Bootstrap pre dizajn. Rozhranie 5.1 má formu webovej stránky ktorá umožňuje správu kolekcií a obrazov registrovaným užívateľom a vychádza z návrhu uvedeného v kapitole 4.

Registrácia užívateľa pre vybranú galériu je voľne prístupná z úvodnej webovej stránky rovnako ako prihlásenie. Každý užívateľ má ale prístup len k kolekciám a obrazom ktoré v systéme sám vytvoril alebo ich do systému nahral. Pre overenie identity užívateľa sú porovnané dáta o užívateľovi v databáze (prihlasovacie meno a hash hesla) s tými ktoré poskytol užívateľ pri pokuse o prihlásenie. Pre prístup k neverejným častiam stránky bol využitý princíp webových relácií ktoré sú spravované serverom a databázovým systémom a zabezpečujú identifikáciu užívateľa. Ich implementácia, mimo vstavaných podporných funkcií frameworku Vapor, pozostáva s štruktúr a funkcií implementovaných v súbore `Curator.swift`. Pre autentifikáciu používateľa je to `UserModelCredentialsAuthenticator:` ktorý porovnáva identifikačné dáta z databáze s zadanými údajmi a `UserModelSessionAuthenticator:` ktorý má na starosti autentifikáciu používateľa pri používaní webovej stránky. Pre prácu s heslami respektíve ich hashovaním je použitá funkcia `BCrypt`[2].

[2]https://en.wikipedia.org/wiki/Bcrypt



Obr. 5.1: Ukážka užívateľského rozhrania webového serveru pre správu obrazov. Jedná sa o úvodnú obrazovku pre prihlásenie alebo registráciu.

V časti webovej stránky venovanej kolekciám má užívateľ možnosť vidieť všetky nám vytvorené kolekcie a ich popis. Webová stránka umožňuje zmenu informácií o už existujúcich kolekciách, vytvorenie novej kolekcie v rámci danej galérie a v prípade, že sa v kolekcií nenachádzajú žiadne obrazy aj jej odstránenie. Po rozkliknutí kolekcie je užívateľovi zobrazený zoznam obrazov v danej kolekcií, spoločne s ich popisom a vybranými dátami. Zobrazenie kompletných dát o obraze alebo ich zmena je sprístupnené po kliknutí na tlačidlo `edit`. Implementovaná je aj možnosť vytvorenia nového obrazu, po zadaní potrebných údajov a nahraní obrazu zo zariadenia na ktorom užívateľ webovú stránku používa. Vymazanie obrazu nieje, na rozdiel od kolekcií, obmedzené žiadnou podmienkou.

Navigácia v rámci webovej stránky je spravovaná preddefinovanými cestami a ich riadiacimi funkciami. Registrácia jednotlivých ciest je realizovaná v súbore `routes.swift`. Tu sa nachádzajú aj pomocné štruktúry, rozpoznateľné pomocou prítomnosti slova `Context` v názve, ktoré majú za úlohu sprostredkovať dáta medzi riadiacimi funkciami, cestami a šablónovacím jazykom Leaf. Riadiace funkcie dodržiavajú jednotnú mennú konvenciu a sú pomenované na základe svojho určenia, napríklad, `registrationHandler()` alebo `collectionDetailHandler()`.

5.2 Mobilná aplikácia

Táto sekcia sa venuje implementácií mobilnej aplikácie určenej na vizualizáciu obrazov v rozšírenej realite pre zariadenia iPhone od Apple Inc. s operačným systémom iOS. Jej implementácia bola realizovaná v programovacom jazyku Swift pomocou frameworkou ako napríklad UIKit či ARKit. Predlohou pre implementáciu používateľského rozhrania bol návrh ktorý je uvedený v kapitole 4. Pre implementáciu funkcií, to bola implementácia vzorovej aplikácie[3] pre prácu s ARKitom od Apple a zoznam definovaných funkčných požiadavkou pre túto mobilnú aplikáciu.

[3]https://developer.apple.com/documentation/arkit/environmental_analysis/placing_objects_and_handling_3d_interaction

Užívateľské rozhranie

Užívateľské rozhranie bolo implementované na základe vytvoreného návrhu. V priebehu vývoja bolo postupne upravované aby odpovedalo zmenám v návrhu ktoré vznikli zapracovaním zistení z testovania mobilnej aplikácie a serveru. Implementácia používateľského rozhrania aplikácie bola realizovaná za pomoci frameworku UIKit.

Framework UIKit poskytuje možnosti pre tvorbu užívateľského rozhrania jednak v grafickom editore alebo priamo v programovacom jazyku Swift. Základná časť užívateľského rozhrania bola preto vytvorená vytvorená v grafickom editore pomocou takzvaných storyboardov a implementácia komplikovanejších prvkov bola realizovaná v programovacom jazyku Swift.

Komunikácia zo serverom

Aplikácia na komunikáciu zo serverom a získavanie dát o galériách, kolekciách a obrazoch používa na serveri implementované API. Využívané sú nasledovné koncové body API rozhrania:

- `/api/galleries` - Získanie dát o existujúcich galériách. Používané na zobrazenie zoznamu galérií z ktorých si užívateľ môže vybrať.
- `/api/paintings` - Získanie dát o všetkých obrazoch v systéme, používané na získanie obrazových dát obrazov ktoré ešte niesu uložené na zariadení.
- `/api/collectionsForGallery/:galleryID` - Získanie dát o kolekciách pre danú galériu. Používané na zobrazenie zoznamu kolekcií na výber, pre užívateľom vybranú galériu.
- `/api/paintingsForCollection/:collectionID` - Získanie dát o všetkých obrazoch pre danú kolekciu. Používané na zobrazenie zoznamu obrazov na výber, pre užívateľom vybranú kolekciu.

Pre prácu s dátami získanými zo serveru používa aplikácia rovnaké štruktúry pre modely ako server, čo je jedna z výhod použitia programovacieho jazyku Swift aj pre implementáciu serverovej časti. Dáta získané cez API sú vo formáte JSON a ich dekódovanie a mapovanie na dané modely prebieha pomocou `JSONDecoder()` respektíve metódy `decode()`.

AR scéna

Pred samotnou inicializáciou AR scény je potrebné ju na základe konkrétnych požiadavkou nakonfigurovať. Táto sekcia sa popisuje výber konfigurácie ktorá je pre implementovanú aplikáciu najvhodnejšia a v druhej časti sa venuje rovnako dôležitému procesu inicializácie, predtým nakonfigurovanej, AR scény.

Konfigurácia

Aplikácia pre plnenie svojej funkcie potrebuje detekovať len vertikálne plochy v prostredí. Detekciu vertikálnych alebo horizontálnych plôch je možné nastaviť pri inicializácií frameworku ARKit. Pre nastavenie konfigurácie je používaná trieda `ARConfiguration`, respektíve, v tomto prípade od nej dediaca trieda `ARWorldTrackingConfiguration`. V tejto konfigurácií sleduje ARKit polohu a pozíciu zariadenia v scéne v šiestich stupňoch voľnosti.

Pre polohu je to rotácia, náklon, otočenie a pre pozíciu je to štandardný troj-osí súradnicový systém. Pri použití konfigurácie `ARWorldTrackingConfiguration` ARKit umožňuje detekciu dopredu známych 2D obrazov, 3D objektov alebo detekciu plôch.

Pre potreby tejto mobilnej aplikácie bola používaná len detekcia vertikálnych plôch. Pre ich korektné detekovanie bolo potrebné nastaviť premennú `planeDetection` v konfigurácií nasledovne: `configuration.planeDetection = [.vertical]`.

Inicializácia

Pre inicializáciu scény v rozšírenej realite za použitia frameworku ARKit je potrebné zavolať metódu `run()` na objekte `session` patriacemu inštancií `ARSCNView`. `ARSCNView` je trieda (view) ktorá umožňuje zobrazenie scény v rozšírenej realite za použitia frameworku SceneKit.

Detekcia vertikálnych plôch

Pre správne fungovanie aplikácie je nutné zaistiť spoľahlivé detekovanie vertikálnych plôch. Samotnú detekciu, podľa zadanej konfigurácie, zabezpečuje framework ARKit. Pre zlepšenie používateľského zážitku a rýchlosti detekcie je žiadúce informovať používateľa o stave detekcie a eventuálne mu zobraziť informácie ako môže detekciu urýchliť. Táto funkcionalita je realizovaná pomocou zobrazenia potrebnej nápovedy poskytovanej ARKitom na obrazovke zariadenia.

Nápoveda pri detekcií

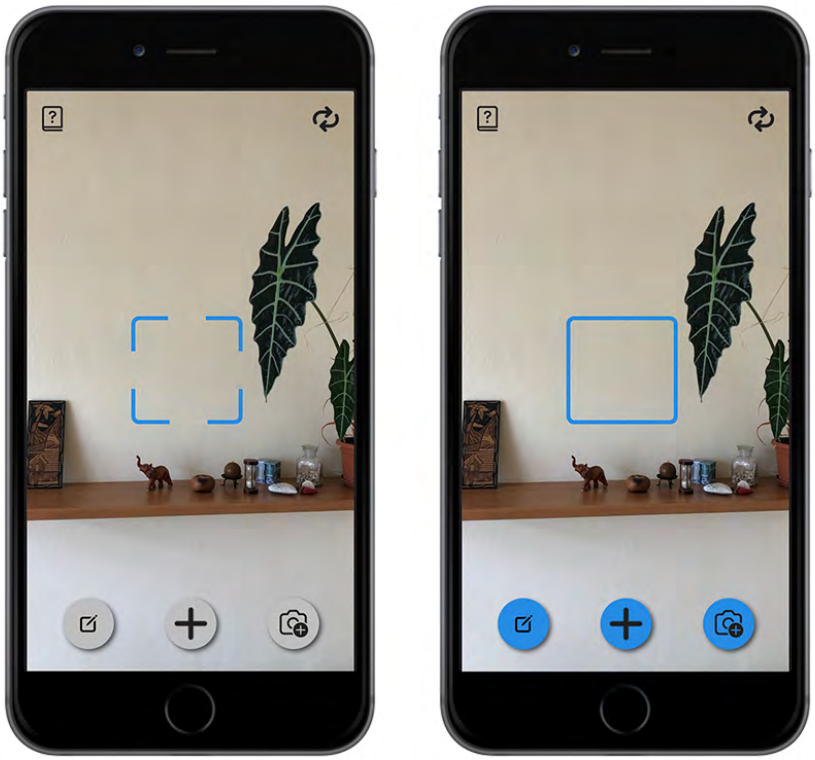
Pre zobrazenie nápovedy pri detekcií vertikálnych plôch na ktoré by bolo možné umiestniť obraz aplikácia využíva prostriedky ktoré ponúka priamo framework ARKit. Jedná sa konkrétne o view `ARCoachingOverlayView` ktoré zobrazuje animované informácie o aktuálnom stave sledovania a akciách ktoré by mal užívateľ vykonať pre detekciu požadovanej plochy. View `ARCoachingOverlayView` je aktivované automaticky po spustení scény s rozšírenou realitou a zobrazuje informácie na základe aktívnej konfigurácie ARKitu. Pokiaľ dôjde k detekcií požadovanej plochy nápoveda sa deaktivuje.

Informovanie užívateľa o detekovaní vhodnej plochy

Nakoľko je potrebné po detekcií vhodnej vertikálnej plochy jej polohu vhodným spôsobom signalizovať užívateľovi, v aplikácia využíva zameriavací štvorec ktorý indikuje miesto kam bude obraz vložený a rovnako aj dostupnosť vhodnej vertikálnej plochy v jeho aktuálnom mieste. Zameriavací štvorec je zobrazený pokiaľ sa v scéne s rozšírenou realitou ešte nenachádza žiadny obraz. Grafickú podobu znázorňuje obrázok návrhu aplikácie 5.2.

Pridávanie objektov do AR scény

Aplikácia pracuje s obrazmi ktoré sú definované ako 3D objekty. Pri pridávaní obrazov do scény s rozšírenou realitou je na model obrazu aplikovaná užívateľom vybraná textúra (obraz). Pri umiestňovaní obrazov na vertikálne plochy aplikácia zohľadňuje vzdialenosť a natočenie danej plochy voči užívateľovi.



Obr. 5.2: Ľavá obrazovka zobrazuje zameriavací štvorec nachádzajúci sa mimo vhodnej detekovanej plochy, vpravo je zobrazený po namierení na vhodnú plochu pre umiestnenie obrazu.

Vytvorenie objektu

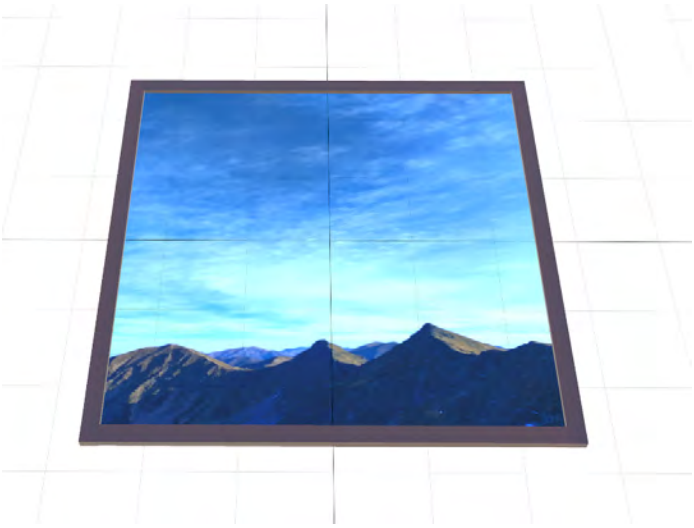
Pre potreby aplikácie boli vytvorené 3D modeli obrazov. Na tieto modeli sa po výbere obrazu užívateľom nanesie požadovaná textúra a je nastavený preddefinovaný druh rámu ktorý sa na obraze nachádza. Obrázok 5.3 zobrazuje jeden z 3D modelov obrazu používaných aplikáciou.

Umiestnenie objektu v AR scéne

Pred samotným umiestnením obrazu do scény sú na základe užívateľom vybraného obrazu pre 3D model nastavené parametre tak aby obraz vložený do scény odpovedal reálnemu. Konkrétne sa jedná o pomer strán, veľkosť obrazu, obraz ktorý má byť zobrazený a 3D modelu je nastavený implicitný rám.

Pre umiestnenie objektu do scény s rozšírenou realitou v aplikácia využíva metódu raycastingu prostredníctvom `ARRaycastQuery`. Počiatočný bod sa nachádza v strede obrazovky. `ARRaycastQuery` vracia vo výsledku pozíciu (hĺbku) v ktorej sa objekt nachádza za predpokladu, že bola vhodná plocha pre jeho umiestnenie zasiahnutá. Rovnako je vo výsledku možné nájsť informáciu o natočení danej plochy ktorá poslúži sa správne natočenie umiestňovaného obrazu.

Transformácia obrazu prebieha aplikovaním získaných informácií o polohe a natočení vybranej virtuálnej vertikálnej plochy voči polohe obrazovky. Z výsledku dodaného prevedenou `ARRaycastQuery` sú získané tieto informácie v premennej `worldTransform` a následne sú aplikované na virtuálny objekt priradením do premennej `simdWorldTransform`.



Obr. 5.3: 3D model obrazu s rámom ktorý je používaný na vizualizáciu v rozšírenej realite v aplikácií.

Realistické zobrazenie obrazu

Pre realistické zobrazenie obrazov, a vo všeobecnosti objektov, ponúka ARKit viaceré možnosti. Jedná sa buď o nastavenia konfigurácie ARKitu alebo o vlastnosti jednotlivých tried ktoré majú na starosti vykresľovanie scény. Pre použitie v implementácií mobilnej aplikácie ktorou sa zaoberá táto práca sú relevantné a využité nasledovné z nich.

- `configuration.environmentTexturing` - Konfiguračný parameter ARKitu ktorý zaistí, že ARKit sa pokúsi z dát zachytených pomocou kamery vytvoriť textúru ktorú následne aplikuje na virtuálne objekty v scéne. Pri zrkadlových povrchoch je možné dosiahnuť takmer reálne pôsobiace odrazy a pri matnejších je táto ilúzia ešte vernejšia. Pre vytvorenie detailnej textúry je ale nutné aby užívateľ zariadením podrobne nasnímal väčšiu časť scény, aj mimo oblasti v ktorej je umiestnený virtuálny objekt. Pre automatické textúrovanie je jeho hodnota nastavená na `.automatic`
- `configuration.isLightEstimationEnabled` - Konfuguračný paramater ktorý nastavuje či sa bude framework ARKit pokúšať z dát o scéne odhadnúť pozíciu zdroja svetla, respektíve svetelné podmienky vrámci scény. Tento odhad je potom použitý na odhad ako veľmi majú byť jednotlivé virtuálne objekty v scéne osvetlené a taktiež môžu byť tieto dáta z odhadu použité na vykreslenie tieňov. Pre povolenie odhadu svetelných podmienok je tento parameter nastavený na `true`. Pre konfiguráciu automatických odhadov svetelných podmienok pre scénu je možné využiť aj vlastnosť triedy `ARSCNView`, `automaticallyUpdatesLighting`.

Pre realistické zobrazenie obrazov je potrebné, na základe imitovaného materiálu, nastaviť aj vlastnosti ich 3D modelov. Medzi najpodstatnejšie patria `Roughness`, `Metalness` a `Ilumination` pre nastavenie hrubosti povrchu, miery lesklosti či reflexívnosti, farby a intenzity osvetlenia povrchu.

Interakcia s objektami v AR scéne

Na základe definovaných funkčných požiadavkou na mobilnú aplikáciu a interakciu s objektami obsahuje implementácia nasledovné tri možnosti interakcie s virtuálnymi obrazmi v scéne s rozšírenou realitou. Ide o zmenu polohy obrazu, zmenu rozmeru obrazu a výber rámu pre daný obraz.

Zmena polohy objektu

Zmena polohy obrazu je podmienená prítomnosťou dostatočne veľkej detekovanej vertikálnej plochy. Pokiaľ by detekovaná plocha neumožňovala presun obrazu alebo by sa užívateľ pokúsil presunúť daný obraz mimo existujúcej plochy, bude v rámci zachovania realistického výzoru scény s rozšírenou realitou obraz ponechaný na svojom mieste respektíve mu bude pohyb obmedzený len v rámci danej vertikálnej plochy. Samotný presun obrazu je realizovaný užívateľom pomocou jednoduchého gesta, kedy stačí na vybranom obraze podržať prst a následne ho presunúť na želané miesto. Táto funkcionalita je implementovaná pomocou `UIGestureRecognizer`.

Konkrétne sú pre detekciu dotyku na obraz a presun obrazu v scéne používané dve triedy dediace od `UIGestureRecognizer`. Pre vybratie obrazu ktorý sa má presúvať je to `UITapGestureRecognizer` a pre presun obrazu v rámci detekovanej vertikálnej plochy `UIPanGestureRecognizer`. `UITapGestureRecognizer` vracia polohu dotyku a je vyžitý pre detekciu, či sa na dotknutom mieste nachádza obraz a pomocou `UIPanGestureRecognizer` je riešený samotný presun obrazu.

Zmena rozmeru objektu

Pri zmene veľkosti obrazu má užívateľ na výber z preddefinovaných rozmerov. Zmena veľkosti neovplyvňuje pomer strán obrazu a ten si teda zachováva svoj tvar. Pre zmenu veľkosti je zobrazovaný virtuálny objekt zmenšovaný alebo zväčšovaný oproti svojej pôvodnej veľkosti uniformnou transformáciou pomocou parametru modelu `scale`, typu `SCNVector3`. Pre dosiahnutie verného zážitku v rozšírenej realite sú obrazy zobrazované v reálnej veľkosti. To je zabezpečené nastavením príslušných parametrov 3D modelu na základe dát o obraze. Obrazovka zobrazujúca možnosti zmeny veľkosti obrazu je zobrazená na obrázku 4.7.

Zmena rámu objektu

Aplikácia umožňuje výber z troch rôznych rámov pre každý z obrazov. Na výber je tiež možnosť zobraziť obraz bez rámu. Jednotlivé rámy imitujú prírodné alebo syntetické materiály využívaných k tvorbe reálnych rámov. To bolo docielené nastavením parametrov pre ich 3D modely. Zmeny sú v použitej textúre či nastavení hodnoty pre lesklosť alebo odrazivosť svetla.

Reset AR scény

V prípadoch kedy framework ARKit nieje schopný pokračovať v sledovaní okolia a zobrazovaní už vytvorenej scény je vhodné dať užívateľovi možnosť scénu resetovať a začať odznovu. K tejto situácií typicky dochádza pri presune používateľa v reálnom prostredí, prípadne pri náhlej zmene osvetlenia v priestoroch kde sa užívateľ nachádza. Pre možnosť pokračovať v používaní aj v takomto prípade aplikácia umožňuje resetovanie scény s rozšírenou realitou pomocou funckie `resetTracking()`.

Kapitola 6

Testovanie a analýza výsledkov

Táto kapitola sa venuje vyhodnoteniu výsledkov a testovaniu mobilnej aplikácie a webového serveru na užívateľoch. Testovanie bolo zamerané na testovanie z hľadiska funkčnosti a následne na testovanie aplikácie používateľmi. V poslednej sekcii kapitoly sú spomenuté možnosti či funkcie ktoré by do aplikácie v budúcnosti mohli byť pridané. Testovanie mobilnej aplikácie prebiehalo na mobilnom zariadení Apple iPhone 8+ a testovanie webového serveru bolo realizované v prostredí internetových prehliadačov Safari a Google Chrome na rôznych zariadeniach.

6.1 Návrh testov

Testovanie z hľadiska funkčnosti bolo navrhnuté na základe požadovanej funkcionality pre mobilnú aplikáciu a webový server. Pri testovaní mobilnej aplikácie boli jednotlivé funkcie rozšírenej reality testované v rôznych priestoroch a na rôznych povrchoch aby sa overila ich spoľahlivosť. Testovaná bola rovnako komunikácia aplikácie zo serverom. Pri webovom serveri bolo testované, či umožňuje realizovať operácie s dátami podľa návrhu. Cieľom tejto časti bolo overiť či mobilná aplikácia a webový server umožňujú použitie na požadovaný účel.

V druhej časti sa testovanie presunulo na užívateľov. Cieľom bolo sledovať interakciu užívateľa s mobilnou aplikáciou a webovým serverom. Pri testovaní aplikácie bolo prvým testovaným faktorom sledovanie odozvy aplikácie na vstupy od používateľa pri vkladaní či interakcií s obrazmi v AR scéne. Ďalším testovaným faktorom bola orientácia používateľa v grafickom rozhraní aplikácie. Cieľom bolo overiť či je užívateľ schopný nájsť a použiť všetky funkcie aplikácie bez detailného návodu a nakoľko je pre užívateľa grafické rozhranie aplikácie prehľadné. Testovanie webového serveru bolo určené na získanie informácií o odozve webového rozhrania serveru a podobne ako testovanie mobilnej aplikácie sledovalo schopnosť užívateľa orientovať sa v rozhraní bez jeho bližšieho predstavenia.

6.2 Testovanie funkčnosti

Testovanie z hľadiska funkčnosti bolo zamerané na overenie požadovanej funkcionality webového serveru a mobilnej aplikácie. Testovanie mobilnej aplikácie bolo rozdelené na dve fázy, a to testovanie spoľahlivosti detekcie vertikálnych rovín či reálneho zobrazenia obrazov a testovanie výslednej aplikácie z hľadiska splnenia funkčných požiadavkou. Testy webového serveru boli zamerané na možnosť realizácie požadovaných operácií s obrazmi a kolekciami.

Testovanie mobilnej aplikácie

Prvá časť testovania bola zameraná na spoľahlivosť detekcie vertikálnych rovín a a samotnej vizualizácie obrazov na nich. Testovanie detekcie vertikálnych plôch prebiehalo pred samotným testovaním užívateľmi výlučne v domácom prostredí vzhľadom na obmedzenia voľného pohybu ktoré boli v platnosti v dôsledku pandémie. Pri testovaní detekcie, boli testované rôzne povrchy, zväčša steny, no pre lepšiu predstavu o prenosnosti bola otestovaná detekcia aj na dverách či vstavaných skriniach.

V druhá časť testovania aplikácie z hľadiska funkčných požiadavkov bola zameraná na ostatné funkcie aplikácie, menovite zmenu veľkosti obrazu, rámu, presúvanie obrazu v rámci scény a reset AR scény. Na obrázku 6.1 je znázornené testovanie zobrazenia obrazu v správnej veľkosti vo virtuálnej realite. Reálnosť zobrazenia bola testovaná porovnaním s materiálmi reálnych objektov a povrchov. Pre rám obrazu sa jednalo o rôzne druhy dreva, pre obraz samotný bol virtuálny povrch porovnávaný s lesklým a matným papierom a jemným plátnom.

Obr. 6.1: Záznam obrazovky zachytáva testovanie nastavenej veľkosti obrazu. Rám je široký 2cm, samotný obraz 28cm. Po umiestnení do virtuálnej reality dané rozmery odpovedajú reálnym vzdialenostiam.

Testovanie webového serveru

Funkcionalita webového severu bola otestovaná na základe predom definovaných požiadavkou na jeho funkcionalitu. Testované teda boli operácie s existujúcimi dátami v systéme ako napríklad zmena údajov o obraze alebo kolekcií a odstránenie daného obrazu či kolekcie. Rovnako bola otestovaná možnosť vytvorenia nových kolekcií a obrazov v systéme. Kontrola jednotlivých testovacích scenárov prebehla zobrazením dát v databáze za pomoci nástroju na zobrazenie dát z PostrgeSQL databázového systému `Postico`[1]

6.3 Testovanie užívateľmi

V tejto časti testovania som sa zameral na prístupnosť jednotlivých funkcií používateľom a celkovú prehľadnosť a prívetivosť používania používateľského rozhrania aplikácie a webového serveru. Testovanie prebiehalo len na obmedzenej vzorke používateľov nakoľko vzhľadom na situáciu okolo pandémie nebolo možné realizovať osobné stretnutia s väčším počtom osôb. Celkovo sa tejto časti testovania zúčastnilo 5 osôb vo vekovom rozmedzí od 19 do 50 rokov.

Testovanie mobilnej aplikácie

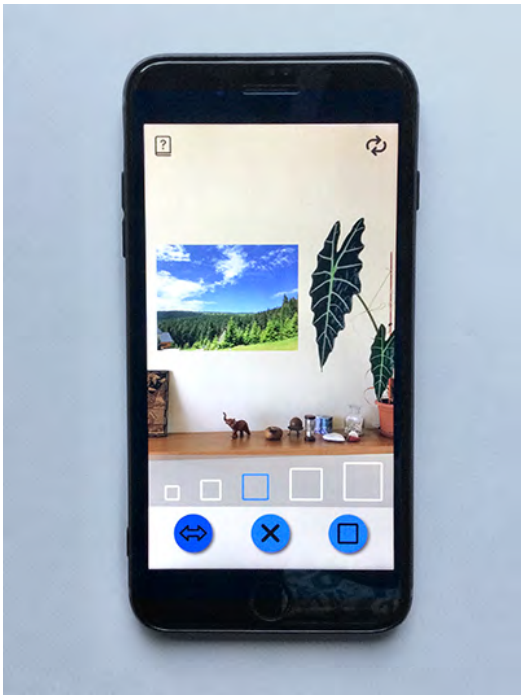
Samotné testovanie mobilnej aplikácie prebehlo v dvoch fázach. V prvej fáze bolo testovanie zamerané na užívateľské rozhranie mobilnej aplikácie pre výber galérie, kolekcie a obrazu. V druhej fáze bola testovaná časť mobilnej aplikácie a jej užívateľke rozhranie umožňujúce prácu s obrazmi v scéne rozšírenej reality.

Testovanie mobilnej aplikácie a jej rozhrania v prvej fáze prebehlo za použitia vytvoreného návrhu a wireframu. Toto umožnila mobilná verzia aplikácie XD od Adobe. Jej použitie znázorňuje obrázok 6.2. Cieľom bolo získať spätnú väzbu od užívateľov ešte pred implementáciou samotnej aplikácie, aby v prípade nepredvídanej chyby v návrhu užívateľského rozhrania pre výber obrazu nebolo nutné celý proces implementácie realizovať odznovu.

V druhej fáze sa testovanie ďalej členilo na dve časti. Prvá časť testovania prebiehala tak, že testujúci používateľ dostal na poskytnutom mobilnom zariadení za úlohu spustiť implementovanú aplikáciu a pomocou nej umiestniť ľubovoľne vybraný obraz na pozíciu ne stene podľa svojho uváženia. Mobilná aplikácia pred týmto testom užívateľom nebola detailnejšie predstavená, keďže úlohou bolo zistiť či a ako rýchlo dokáže užívateľ samostatne vykonať danú úlohu.

V druhej časti bolo užívateľom opäť poskytnuté zariadenie s nainštalovanou aplikáciou. Tentokrát však používatelia nemali presne definovanú úlohu ale boli inštruovaní aby si v aplikácií vyskúšali všetky funkcie a pokúsili sa vytvoriť AR scénu s obrazom podľa svojho vlastného uváženia. Po vytvorení danej scény boli používatelia dotazovaní na ich dojem z užívateľského rozhrania aplikácie. Zámerom bolo získať informácie o tom ktoré prvky sú pre užívateľa dobre použiteľné a ktoré naopak nepôsobia logicky alebo sťažujú prácu s mobilnou aplikáciou.

[1] https://eggerapps.at/postico/



Obr. 6.2: Návrh aplikácie zobrazený v mobilnej aplikácií XD od Adobe. Na prvý pohľad je nerozoznateľný od reálnej implementácie.

Testovanie webového serveru

Testovanie rozhrania webového serveru prebiehalo, na rozdiel od testovania mobilnej aplikácie, za použitia implementovanej verzie nakoľko prípadné zmeny vychádzajúce zo zistených nedostatkov bolo možné realizovať v implementácií jednoduchšie. Testovanie webového serveru bolo vzhľadom na obmedzenia spôsobené pandémiou obmedzené na nižší počet osôb. Vzorka používateľov teda zahŕňala 4 osoby vo veku od 20 do 43 rokov.

Samotnému testovaniu prebiehalo krátke predstavenie webového rozhrania užívateľom a ich oboznámením s účelom použitia webového serveru. Následne boli používatelia inštruovaní aby vykonali podporované akcie, a teda zaregistrovali sa na webový portál, prihlásili sa do systému, vytvorili novú kolekciu a obrazy, zmenili informácie o vybranej kolekcií či vybranom obraze a pokúsili sa danú kolekciu alebo obraz zo systému odstrániť.

Zámerom tejto časti testovania bolo získanie spätnej väzby na intuitívnosť a prehľadnosť používania webového rozhrania serveru. Po vykonaní vyššie uvedených akcií boli užívatelia dotazovaní na prvky ktoré pre nich boli nelogické, neprehľadné alebo im inak sťažovali orientáciu v systéme a jeho následné používanie.

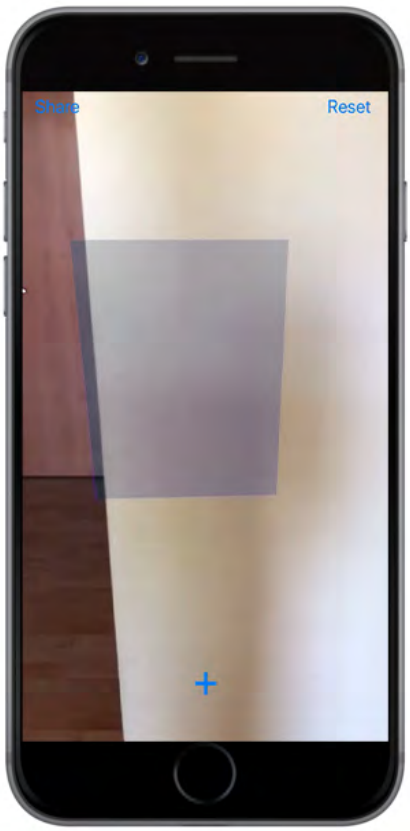
6.4 Vyhodnotenie

Celkovo dopadlo testovanie aplikácie a webového serveru uspokojivo. Neboli zistené žiadne závažné nedostatky či chyby vo funkcionalite a užívatelia aplikáciu a webový server po zapracovaní pripomienok hodnotili kladne. Zhodnotenie jednotlivých častí testovania nasleduje nižšie.

Vyhodnotenie testovania mobilnej aplikácie

Výsledky prvej časti testovania boli celkovo uspokojivé. Na povrchoch s výraznejším vzorom alebo textúrou aplikácia nemala problém úspešne detekovať vertikálne roviny a umiestniť na nich obraz. Pri jednofarebných povrchoch bez výraznejších prvkov či textúr bola detekcia pomalšia ale pri dostatočnom priblížení k danému povrchu sa aplikácií podarilo detekovať vertikálnu plochu aj na týchto povrchoch. Detekovanú plochu na jednofarebnej stene zobrazuje obrázok 6.3. Pri povrchoch s výraznejšou textúrou a nejednofarebných povrchoch prebiehala detekcia vertikálnych plôch bezproblémovo. Zistenia z tejto časti testovania boli zahrnuté do nápovedy ktorá sa pri inicializácií AR scény zobrazuje užívateľovi v aplikácií.

Druhá časť testovania dopadla rovnako úspešne. Mobilná aplikácia nemala problém pri komunikácií zo serverom prostredníctvom API, zo zmenou veľkosti zobrazovaného obrazu či resetovaním AR scény. Takisto je zobrazenie materiálu obrazu dostatočne podobné realite.



Obr. 6.3: Ukážka testu detekcie vertikálnej roviny na jednofarebnej stene bez výraznej textúry. Detekovaná rovina je v AR scéne zobrazovaná svetlomodrou farbou. Zobrazená podoba aplikácie neodpovedá v konečnej verzií, jedná sa o aplikáciu s prototypom užívateľského rozhrania nakoľko najskôr bola implementovaná funkčná stránka aplikácie a až následne užívateľské rozhranie.

Testovanie aplikácie, respektíve jej prototypu pomocou aplikácie XD od Adobe, používateľmi ukázalo, že užívatelia v mobilnej aplikácií dokážu vykonať zadané úlohy. Väčšina z nich nemala problém v prvej časti testu umiestniť obraz do AR scény a s výnimkou jedného boli užívatelia schopný v rozumnom čase samostatne danú úlohu splniť. Získané dáta zobrazuje tabuľka na obrázku 6.4.

Testovanie mobilnej aplikácie užívateľmi

Používateľ č.	Potrebný čas [s]	Nápoveda
1	14	Nie
2	15	Nie
3	21	Nie
4	40	Áno
5	18	Nie

Obr. 6.4: Tabuľka zobrazuje informácie získané v prvej časti užívateľského testovania. Testovacou úlohou bolo umiestnenie obrazu do scény bez predchádzajúceho oboznámenia užívateľov s aplikáciou.

Pri získavaní spätnej väzby na užívateľské rozhranie boli názory rôzne. Kompletné získané dáta zobrazuje graf na obrázku 6.5. Najviac sťažností sa týkalo neprehľadnej navigácie v užívateľskom rozhraní pri výberu galérie, kolekcie a obrazu. To bolo spôsobené viacerými faktormi no tento nedostatok bol vyriešený postupným iteračným upravovaním používateľského rozhrania aplikácie pre výber obrazu na základe získanej spätnej väzby. Došlo tak k zjednodušeniu a sprehľadneniu navigácie v rámci výberu galérie, jej kolekcie a následne konkrétneho obrazu. V prípade užívateľského rozhrania aplikácie v časti pracujúcej s rozšírenou realitou došlo len k malým úpravám zobrazovaných prvkov na ktoré sa užívatelia sťažovali, tak aby pôsobili menej rušivo.

Obr. 6.5: Spätná väzba získaná od užívateľov počas testovania užívateľského rozhrania. Zobrazený graf znázorňuje najčastejšie sťažnosti užívateľov a ich početnosť.

Vyhodnotenie testovania webového serveru

Výsledky prvej časti testovania webového serveru splnili očakávania. Funkcionalita ktorú mal server na základe návrhu poskytovať bola implementovaná a funkčná. Práca s databázou prostredníctvom webového rozhrania fungovala, bolo možné vykonávať zmeny v už

existujúcich dátach, pridávať dáta nové a nežiadúce dáta vymazať. Webové API poskytovalo všetky potrebné údaje a dostupné boli aj obrazy uložené vo verejnom priečinku.

Časť testovania webového serveru a jeho používateľského rozhrania realizovaná za pomoci používateľov dopadla rovnako uspokojivo. Všetci používatelia dokázali splniť zadané testovacie úlohy bez potreby dodatočnej nápovedy či usmernenia. Zo získanej spätnej väzby od užívateľov bolo zrejmé že používateľské rozhranie webového serveru je na svoju úlohu vhodné no našli sa drobné pripomienky na umiestenie niektorých prvkov používateľského rozhrania. Tieto pripomienky boli zvážené a na ich základe došlo k miernemu upraveniu implementácie používateľského rozhrania serveru v jeho výslednej iterácií.

6.5 Možné vylepšenia

Systém mobilnej aplikácie a webového serveru splnil požiadavky ktoré boli zadané ale počas jej tvorby boli zistené a objavené zaujímavé nápady či technológie o ktoré by do budúcna mohla byť aplikácia rozšírená.

V prvom rade sa jedná o editor rámov pre obrazy, kedy by bolo užívateľovi umožnené vytvorenie vlastného rámu. Medzi zaujímavé vlastnosti ktoré by mohli na ráme byť prispôsobované by bolo relevantné zaradiť vlastný tvar, hrúbku či materiál z ktorého je rám zhotovený.

Ďalším zaujímavým rozšírením pre tento systém mobilnej aplikácie a webového serveru by bola integrácia neurónovej siete, na strane aplikácie, ktorá by umožnila transformáciu umeleckého štýlu zobrazovaného obrazu.

Kapitola 7

Záver

Cieľom práce bolo získanie vedomostí o systémoch typu klient-server, rozšírenej realite, jej minulosti, súčastnosti a platformách umožňujúcich prácu s ňou, zo zameraním sa na zariadeniach od spoločnosti Apple s mobilným operačným systémom iOS. Na základe týchto znalostí bola realizovaná následná implementácia mobilnej aplikácie ktorá by umožňovala vizualizáciu obrazov na vertikálnych plochách v rozšírenej realite a s ňou spolupracujúceho webového serveru pre správu obrazov.

Výsledkom práce je implementácia mobilnej aplikácie umožňujúcej vizualizáciu obrazov na vertikálnych plochách v rozšírenej realite a webový server pre ich správu. Aplikácia k práci s rozšírenou realitou využíva framework ARKit. Vytvorená aplikácia umožňuje okrem vizualizácie aj prispôsobenie vizualizovaných obrazov, konkrétne zmenu ich veľkosti alebo prispôsobenie rámu obrazu. Pre dosiahnutie realistickej podoby scény vytvorenej v rozšírenej realite za pomoci aplikácie sa obrazy zobrazujú vo veľkosti ktorá odpovedá ich skutočným rozmerom a ich vzhľad imituje reálne obrazy. Webový server umožňuje pridávanie a správu obrazov, ktoré mobilnej aplikácií sprístupňuje pomocou API.

Výsledný systém mobilnej aplikácie a webového serveru bol testovaný z hľadiska funkčnosti a rovnako bol podrobený testovaniu niekoľkými používateľmi. Z hľadiska funkčnosti aplikácia aj server splnili požiadavky avšak testovanie užívateľmi odhalilo niektoré nedostatky ich užívateľských rozhraní. Opodstatnené pripomienky od užívateľov boli postupne v ďalších iteráciách vývoja adresované a nedostatky užívateľského rozhrania odstraňované. Vo výsledku aplikácia užívateľov zaujala a získala pozitívnu spätnú väzbu.

Požiadavky vyplývajúce zo zadania ako aj tie ktoré boli stanovené pri návrhu aplikácie a webového serveru sa podarilo dodržať a splniť. Pri práci som získal poznatky či nápady ktoré by do budúcna mohli systém vylepšiť a ktoré sú zhrnuté v sekcií 6.5. Prácu považujem za prínosnú z hľadiska prehĺbenia znalostí o vývoji mobilných aplikácií pre mobilný operačný systém iOS, vývoji webových serverov a aplikácií sa použitia programovacieho jazyku Swift a možnostiach využitia rozšírenej reality.

Literatúra

[1] *Best mobile AR games to prove that technology is awesome* [online]. 2019 [cit. 2020-12-16]. Dostupné z: https://www.gamesradar.com/best-mobile-ar-games/.

[2] *10 Real Use Cases for Augmented Reality* [online]. 2018 [cit. 2020-12-28]. Dostupné z: https://www.inc.com/james-paine/10-real-use-cases-for-augmented-reality.html.

[3] *ARConfiguration ARKit | Apple Developer Documentation* [online]. 2020 [cit. 2021-02-03]. Dostupné z: https://developer.apple.com/documentation/arkit/arconfiguration.

[4] *ARCore Fundamental Concepts* [online]. 2018 [cit. 2021-02-06]. Dostupné z: https://developers.google.com/ar/discover/concepts.

[5] *Beautiful ARCore examples that show augmented reality apps are the next big thing* [online]. 2020 [cit. 2020-11-15]. Dostupné z: https://www.androidb.com/2017/09/beautiful-arcore-examples-that-show-augmented-reality-apps-are-the-next-big-thing/.

[6] *ARKit | Apple Developer Documentation* [online]. 2020 [cit. 2021-02-03]. Dostupné z: https://developer.apple.com/documentation/arkit.

[7] *Introducing ARKit: Augmented Reality for iOS* [online]. 2017 [cit. 2020-12-12]. Dostupné z: https://developer.apple.com/videos/play/wwdc2017/602/.

[8] *ARKit - Managing Session Lifecycle and Tracking Quality* [online]. 2020 [cit. 2021-03-05]. Dostupné z: https://developer.apple.com/documentation/arkit/managing_session_lifecycle_and_tracking_quality.

[9] Baum, L. F. *The Master Key: An Electrical Fairy Tale*. 1. vyd. Theophania Publishing, 2011. ISBN 9781770832671.

[10] *Brainapse* [online]. 2018 [cit. 2020-11-3]. Dostupné z: https://www.genengnews.com/magazine/december-1-2018-vol-38-no-21/brainapse/.

[11] *8 Infamous Tech Flops That Were Way Ahead of Their Time* [online]. 2017 [cit. 2020-11-8]. Dostupné z: http://hiresaudiocentral.com/8-infamous-tech-flops-that-were-way-ahead-of-their-time/.

[12] *Growing use of advance technology in Healthcare industry to boost the global Augmented (AR) and Virtual Reality (VR) in Healthcare market 2019 to 2025* [online]. 2019 [cit. 2020-12-18]. Dostupné z: https://www.openpr.com/news/1771641/growing-use-of-advance-technology-in-

healthcare-industry-to-boost-the-global-augmented-ar-and-virtual-reality-vr-in-healthcare-market-2019-to-2025-microsoft-mindmaze-google-psious-daqri-augmedix-medical-realities-firsthand-technology.html.

[13] *IKEA Place App* [online]. 2017 [cit. 2021-12-04]. Dostupné z: https://www.ikea.com/ch/en/customer-service/mobile-apps/ikea-place-app-pub0bab12b1.

[14] Inc., A. *Presentation Slides* [online]. 2017 [cit. 2021-03-20]. Dostupné z: https://devstreaming-cdn.apple.com/videos/wwdc/2017/602pxa6f2vw71ze/602/602_introducing_arkit_augmented_reality_for_ios.pdf?dl=1.

[15] Kipper, G. a Rampolla, J. *Augmented Reality: An Emerging Technologies Guide to AR*. 1. vyd. Elsevier Science and Technology Books, 2012. ISBN 9781597497336.

[16] *Kitura framwork* [online]. 2021 [cit. 2021-04-04]. Dostupné z: https://www.kitura.dev.

[17] *How Augmented Reality Will Disrupt The Manufacturing Industry* [online]. 2019 [cit. 2020-12-18]. Dostupné z: https://blog.thomasnet.com/augmented-reality-manufacturing.

[18] Peddie, J. *Augmented Reality*. 1. vyd. Springer Verlag, 2017. ISBN 3319545019.

[19] *Occluding Virtual Content with People* [online]. 2018 [cit. 2021-04-12]. Dostupné z: https://developer.apple.com/documentation/arkit/occluding_virtual_content_with_people.

[20] *Server side Swift - Perfect | Perfect.org* [online]. 2021 [cit. 2021-04-04]. Dostupné z: https://www.perfect.org.

[21] *ReBlink* [online]. 2020 [cit. 2021-02-12]. Dostupné z: https://ago.ca/exhibitions/reblink.

[22] *Swift* [online]. 2021 [cit. 2020-12-04]. Dostupné z: https://developer.apple.com/swift/.

[23] *Vapor* [online]. 2021 [cit. 2021-04-03]. Dostupné z: https://vapor.codes.

[24] *Vapor 3.0.0 released* [online]. 2018 [cit. 2021-03-18]. Dostupné z: https://medium.com/@codevapor/vapor-3-0-0-released-8356fa619a5d.

[25] *Vapor Docs* [online]. 2021 [cit. 2021-04-03]. Dostupné z: https://docs.vapor.codes/4.0/.

[26] *Vuforia overview* [online]. 2020 [cit. 2020-12-13]. Dostupné z: https://library.vuforia.com/content/vuforia-library/en/features/overview.html.

Príloha A

Plagát

Obr. A.1: Plagát prezentujúci výslednú mobilnú aplikáciu.

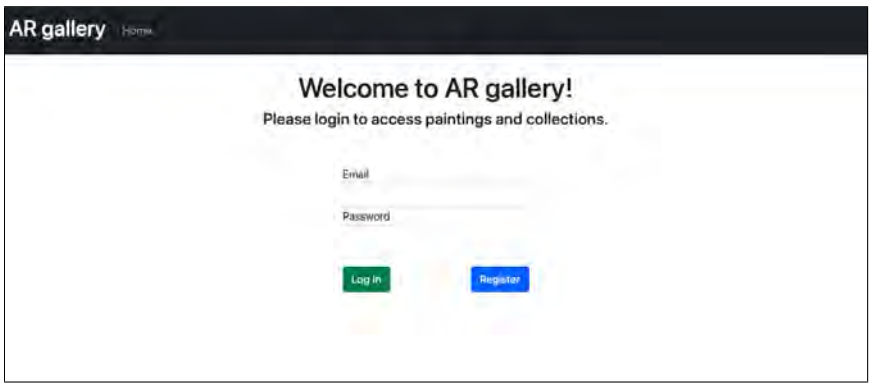
Príloha B

Obsah priloženého pamäťového média

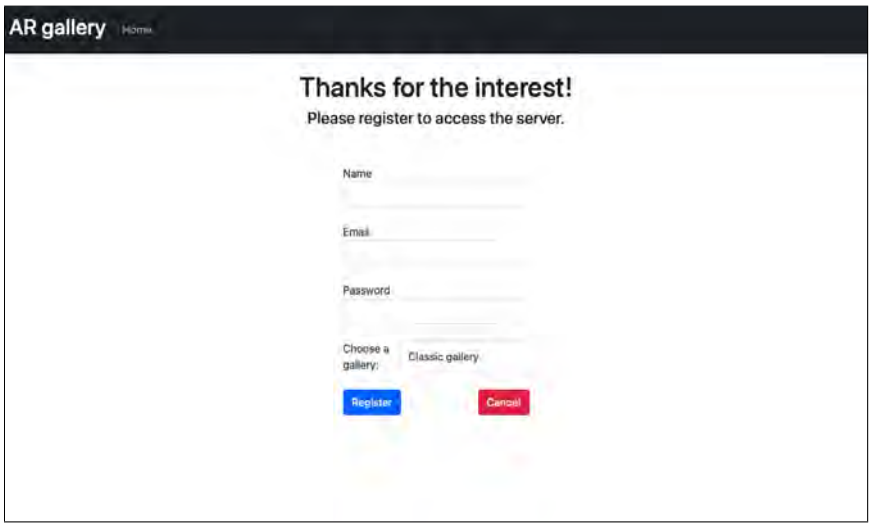
- **src_tex** - zdrojové súbory technickej správy
- **src** - zdrojové súbory
  - **mobile_app** - zdrojové súbory pre mobilnú aplikáciu
  - **web_server** - zdrojové súbory pre webový server
- **media** - prezentačné video a plagát
- ***bp.pdf*** - text bakalárskej práce
- ***readme.txt*** - informácie o bakalárskej práci a návod na spustenie

Príloha C

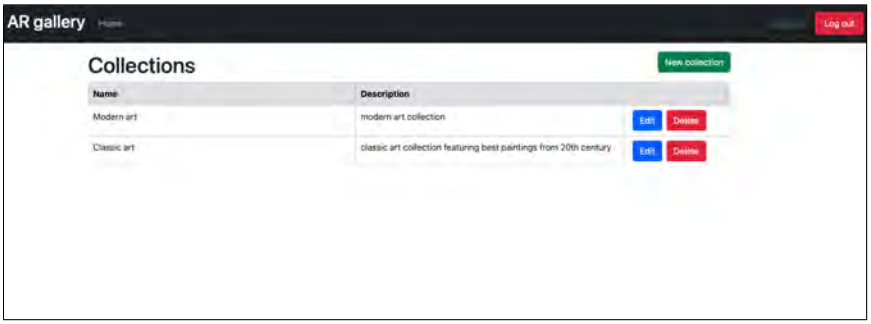
Obrazovky webového serveru



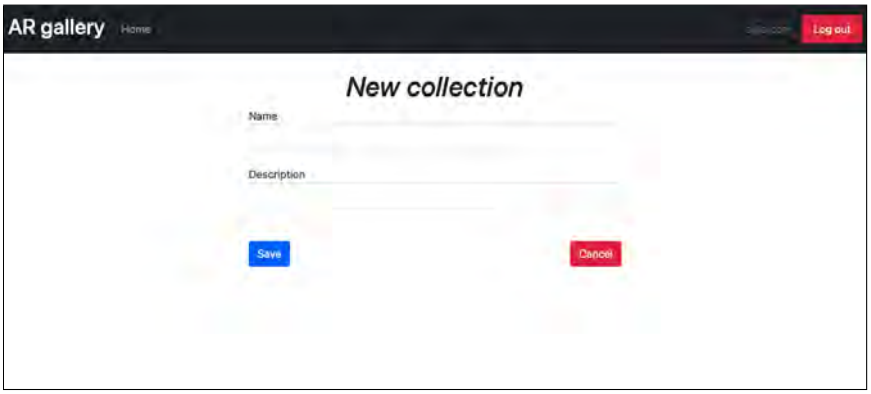
Obr. C.1: Prihlasovacia obrazovka.



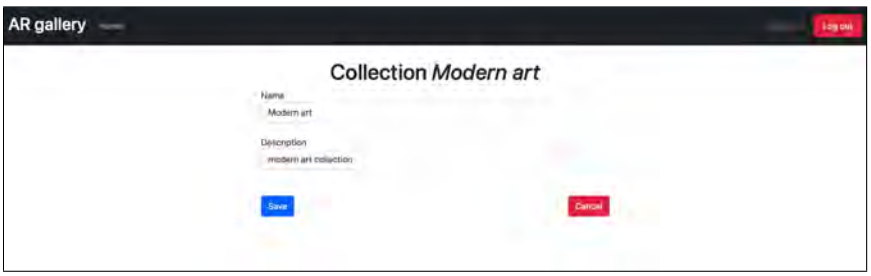
Obr. C.2: Obrazovka pre registráciu.



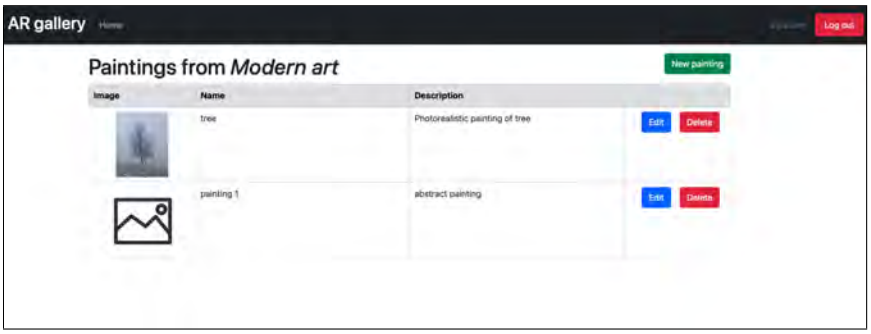
Obr. C.3: Obrazovka zobrazujúca dostupné kolekcie pre daného užívateľa.



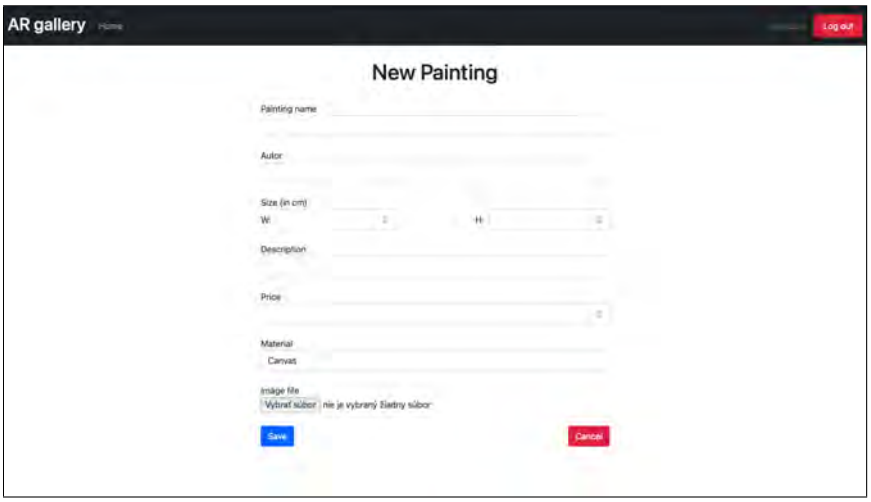
Obr. C.4: Obrazovka vytvárania novej kolekcie.



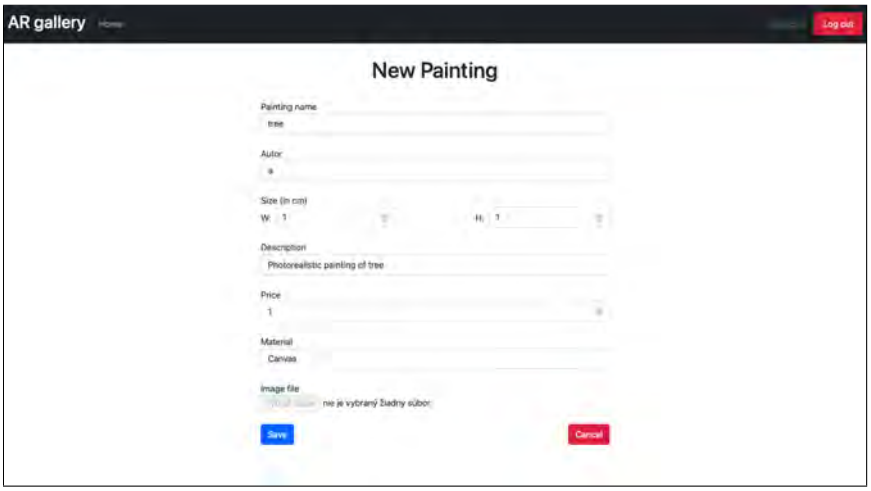
Obr. C.5: Obrazovka upravovania informácií o existujúcej kolekcií.



Obr. C.6: Obrazovka zobrazujúca dostupné obrazy pre danú kolekciu.



Obr. C.7: Obrazovka vytvárania nového obrazu.



Obr. C.8: Obrazovka upravovania informácií o existujúcom obraze.